CBRecSys 2016. New Trends on Content-Based Recommender Systems

Proceedings of the 3rd Workshop on New Trends on Content-Based Recommender Systems co-located with 10th ACM Conference on Recommender Systems (RecSys 2016)

Bogers, Toine; Lops, Pasquale; Koolen, Marijn; Musto, Cataldo; Semeraro, Giovanni

Proceedings of the

CBRecSys 2016

3rd Workshop on New Trends in Content-based Recommender Systems

September 16, 2016

In conjunction with the
10th ACM Conference on Recommender Systems
Boston, MA, USA

Edited by

Toine Bogers, Pasquale Lops, Marijn Koolen,
Cataldo Musto, Giovanni Semeraro

Preface

While content-based recommendation has been applied successfully in many different domains, it has not seen the same level of attention as collaborative filtering techniques have. In recent years, competitions like the Netflix Prize, CAMRA, and the Yahoo! Music KDD Cup 2011 have spurred on advances in collaborative filtering and how to utilize ratings and usage data. However, there are many domains where content and metadata play a key role, either *in addition to* or *instead of* ratings and implicit usage data. For some domains, such as movies, the relationship between content and usage data has seen thorough investigation already, but for many other domains, such as books, news, scientific articles, and Web pages we do not know *if* and *how* these data sources should be combined to provide the best recommendation performance.

The CBRecSys workshop series aims to address this by providing a dedicated venue for papers dedicated to all aspects of content-based recommendation. The first edition in Silicon Valley in 2014, and the second one in Vienna were a big success.

For the third edition, CBRecSys 2016, we once again issued a call for papers asking for submissions of novel research papers addressing recommendation in domains where textual content is abundant (e.g., books, news, scientific articles, jobs, educational resources, Web pages, etc.) as well as dedicated comparisons of content-based techniques with collaborative filtering in different domains. Other relevant topics included opinion mining for text/book recommendation, semantic recommendation, content-based recommendation to alleviate cold-start problems, deep learning for content representation, as well as serendipity, diversity and cross-domain recommendation.

Each submission was rewiewed by three members of the program committee consisting of experts in the field of recommender systems and information retrieval. We selected 9 papers from the 14 submissions for presentation at the workshop.

We are also happy to have Prof. Barry Smyth of the University College Dublin and Prof. Bamshad Mobasher of the DePaul Univesity as keynote speakers.

We thank all PC members, our keynote speakers, as well as authors of accepted papers for making CBRecSys 2016 possible. We hope you will enjoy the workshop!

Toine Bogers, Pasquale Lops, Marijn Koolen, Cataldo Musto, Giovanni Semeraro

August 2016

Organizing Committee

Workshop Co-Chairs

Toine Bogers, Aalborg University Copenhagen
Marijn Koolen, Netherlands Institute of Sound and Vision
Cataldo Musto, University of Bari "Aldo Moro"
Pasquale Lops, University of Bari "Aldo Moro"
Giovanni Semeraro, University of Bari "Aldo Moro"

Program Committee

Jon Atle Gulla, Norwegian University of Science and Technology
Shlomo Berkovsky, NICTA
Ludovico Boratto, University of Cagliari
Robin Burke, DePaul University
Iván Cantador, Universidad Autónoma de Madrid
Federica Cena, Universita' degli Studi di Torino
Paolo Cremonesi, Politecnico de Milano
Marco de Gemmis, University of Bari
Ernesto William De Luca, Potsdam University of Applied Sciences
Tommaso Di Noia, Politecnico di Bari
Peter Dolog, Aalborg University
Fabio Gasparetti, Roma Tre University
Cristina Gena, Universita' degli Studi di Torino
Frank Hopfgartner, University of Glasgow
Juan F. Huete, Universidad de Granada
Jaap Kamps, University of Amsterdam
Silvia Likavec, Universita' degli Studi di Torino
Babak Loni, Delft University of Technology
Fedelucio Narducci, University of Bari
Casper Petersen, University of Copenhagen
Shaghayegh Sahebi, University of Pittsburgh
Alan Said, University of Skövde
Marco Tkalčič, Free University of Bozen-Bolzano
Bei Yu, Syracuse University

Table of Contents

Invited presentations

Accepted papers

From Reviews to Recommendations

Barry Smyth
University College Dublin
barry.smyth@ucd.ie

Abstract

Recommender systems are now a familiar part of the digital landscape helping us to choose which movies to watch and books to read. They guide us about where to stay and eat when we travel. They help us to keep in touch with friends and may even influence our choice of a mate. To do this recommender systems require data. Lots of data.

In the early years this data came in the form of our online transactions and item ratings. More recently recommendations have been influenced by our social networks, the connections that link us, and the things that we share with others. Today there is a new form of data that has the potential to drive recommender systems of the future: user-generated reviews. Reviews are now a routine part of how we make decisions, large and small. Most of us wouldn't dream of booking a hotel without first checking out its reviews and companies like TripAdvisor and Yelp have build billion dollar enterprises on the opinions of millions of people.

In this talk we will discuss the role of user generated reviews in a new generation of recommender systems and some of the ways that opinions can be leveraged to better understand users and generate new forms of recommendations. We will focus on how opinion mining techniques can be used to extract features and sentiment from unstructured review text and ways to use this information in recommendation ranking and explanation.

Context v. Content:
The role of semantic and social knowledge in context-aware recommendation

Bamshad Mobasher
DePaul University
mobasher@cs.depaul.edu

Abstract

Context aware recommender systems go beyond the traditional personalized recommendation models by incorporating a form of situational awareness. They provide recommendations that not only correspond to a user's preference profile, but that are also tailored to a given situation or context.

In many domains, explicit contextual factors and their values may not be known to the system a priori, but may need to be learned or inferred in the course of user's interaction with the system. Moreover, the contextual state of a user can be dynamic and change during that interaction. In such systems, semantic knowledge about the domain, content features extracted from items, and social annotations representing user attitudes and interests can be a source of additional contextual information that can be used to effectively for inferring contexts and adapting to contextual changes.

In this talk we focus on the role of this type of semantic and social knowledge as part of the design of hybrid context-aware recommender systems. We will explore several case studies that demonstrate the interaction between context and semantic or social knowledge. In particular, we will look at an approach where user profiles are represented as mixtures of the latent topics allowing for a unified model of users, items, and the meta-data associated with contexts; an approach where contextual information in obtained by mining social annotations or other textual features associated with a user's current situation and used in combination with user preference histories to compute a utility function over the set of items; and an approach that emphasizes the role of a domain ontology in the form of a concept hierarchy as an integral part of a user's evolving contextual profile.

Combining Content-based and Collaborative Filtering for Personalized Sports News Recommendations

Philip Lenhart
Department of Informatics
Technical University of Munich
Boltzmannstr. 3, 85748 Garching, Germany
philip.lenhart@in.tum.de

Daniel Herzog
Department of Informatics
Technical University of Munich
Boltzmannstr. 3, 85748 Garching, Germany
herzogd@in.tum.de

ABSTRACT

Sports news are a special case in the field of news recommendations as readers often come with a strong emotional attachment to selected sports, teams or players. Furthermore, the interest in a topic can suddenly change if, for example, an important sports event is taking place. In this work, we present a hybrid sports news recommender system that combines content-based recommendations with collaborative filtering. We developed a recommender dashboard and integrated it into the Sport1.de website. In a user study, we evaluated our solution. Results show that a pure content-based approach delivers accurate news recommendations and the users confirm our recommender dashboard a high usability. Nevertheless, the collaborative filtering component of our hybrid approach is necessary to increase the diversity of the recommendations and to recommend older articles if they are of special importance to the user.

CCS Concepts

•**Information systems** → **Recommender systems;**

Keywords

Recommender System; Sports News; Content-based; Collaborative Filtering; Hybrid

1. INTRODUCTION

Recommender systems (RSs) suggest items like movies, songs or points of interest based on the user's preferences. Traditional RSs have to face some challenges when recommending such items. One of the most common problems is the cold-start problem [1]. News items without any ratings cannot be recommended while new users who did not share their preferences with the RS yet cannot receive any personalized recommendations. When recommending news, recency plays a critical role [11]. News have to be up-to-date but sometimes older articles are important if there is a connection to current events. Sports news represent only one category of news but they complicate the news recommendation process. People interested in sports are often characterized by a strong emotional attachment to selected sports, teams or players. With regard to recommendations, a user could be in favor of a lot of news about one team while she or he absolutely wants to avoid any information about a rival. Furthermore, the user's interest in a topic can suddenly change. For example, during the Fifa World Cup, even some people who are not interested at all in soccer want to be kept up-to-date with regard to current results.

In this work, we want to examine how well content-based RSs work for recommending sports news. In addition, we extend our RS by collaborative filtering. We develop a recommender dashboard and integrate it into the website of the German television channel and Internet portal Sport1[1]. We evaluate both algorithms and the usability of our prototype in a user study.

This paper is structured as follows: in Section 2 we present related work and highlight our contribution to the current state of research in content-based and hybrid news RSs. We explain how we combine a content-based and a collaborative filtering component to a hybrid sports news RS in Section 3. Our development is evaluated in a user study. The results of this study are summarized in Section 4. This work ends with a conclusion and an outlook on future work.

2. RELATED WORK

Different approaches try to tackle the problem of personalized news recommendations. One of the first news RSs was developed and evaluated by the GroupLens project [9]. The researchers used collaborative filtering to provide personalized recommendations. A seven-week trial showed that their predictions are meaningful and valuable to users. Furthermore, they found out that users value such predictions for news because in the experiment, the participants tended to read highly rated articles more than less highly rated articles. Liu et al [10] developed a news RS based on profiles learned from user activity in Google News. They modeled the user's interests by observing her or his past click history and combined it with the local news trend. Compared with an existing collaborative filtering method, their combined method improved the quality of the recommendations and attracted more frequent visits to the Google News website.

Using article keywords to build user profiles for news recommendations has already been researched. The Personalized Information Network (PIN) creates user profiles by so

[1]http://www.sport1.de/

CBRecSys 2016, September 16, 2016, Boston, MA, USA.

called interest terms which consist of one or more keywords [15]. Experiments show that PIN is able to deliver personalized news recommendations on-the-fly.

Some researchers used hybrid RS combining different techniques to suggest news articles. Claypool et al. [7] developed P-Tango, an online newspaper combining the strengths of content-based and collaborative filtering. News@hand is a system that makes use of semantic-based technologies to recommend news [5]. It creates ontology-based item descriptions and user profiles to provide personalized, context-aware, group-oriented and multi-facet recommendations. Its hybrid models allow overcoming some limitations of traditional RS techniques such as the cold-start problem and enables recommendations for grey sheeps, i.e. users whose preferences do not consistently agree or disagree with any group of people [7]. The authors evaluated the personalized and context-aware recommendation models in an experiment with 16 participants. Results showed that the combination of both models plus their semantic extension provides the best results [6]. De Pessemier et al. [8] used an hybrid approach to recommend news of different sources. Their approach combines a search engine as a content-based approach with collaborative filtering and uses implicit feedback to determine if the user is interested in a certain topic. The recommendations are presented in a web application optimized for mobile devices.

Asikin and Wörndl [2] presented approaches for recommending news article by using spatial variables such as geographic coordinates or the name and physical character of a location. Their goal was to to deliver serendipitous recommendation while improving the user satisfaction. A user study showed that their approaches deliver news recommendations that are more surprising than a baseline algorithm but still favored by the users.

To the best of our knowledge, no research focusing on the special case of sports news has been done. In this work we want to show how sports news can be recommended in a content-based approach. In addition, we extend this RS by a collaborative filtering component. In a user study, we evaluate both approaches to find out if the hybrid algorithm improves the recommendations. We show how sports news can be suggested to real users by developing and testing a fully working recommender dashboard which can be integrated into existing webpages.

3. DEVELOPMENT OF A PERSONALIZED SPORTS NEWS RECOMMENDER SYSTEM

This section explains the algorithms we used in our RS, but also illustrates the user profile modeling that is needed to provide personalized recommendations. Finally, the prototype is shown to point out how our concepts are implemented on a website.

3.1 User Profile and Preference Elicitation

The user's preferences with regard to sports news are expressed by keywords of articles that she or he is reading. Each article of our recommendation database is characterized by five to ten keywords which are automatically generated by analyzing the article's text. We are storing a list of keywords and how often each keyword occurs in articles the user has read. The more articles the user is reading, the better the recommendations are optimized with regard to the user's preferences. In our first prototype the counter for each present keyword is incremented by one when the user reads the article containing this keyword. In future works, the keywords in an article could be weighted according to the relevance and importance of the keyword to the article.

The new user problem affects every user who did not read an article yet. As explained, sports news differ from other kinds of news in the emotional attachment to selected sports, teams or players. We use this finding to overcome the new user problem. Before starting the recommendation process, the user can specify her or his favorite sport and team. News can then be recommended based on this selection and will improve when the user is reading articles, thus providing implicit feedback.

3.2 Content-based Sports News Recommendations

Content-based recommender suggest items that are similar to items the user liked in the past [1]. Since the user profile uses weighted keywords, we use vector representations of the profile and the articles to calculate the similarity between two articles.

One of the most important things for a news RS is to provide articles that are not dated. Especially in the sports news domain the environment is fast changing and usually the user is not interested in news about a sports event or her or his favorite team that are not up-to-date. The main challenge for us was to determine how old sports news can be before they are not considered for recommendation anymore. For our content-based RS we only take news into account that are not older than three days. Besides only providing relevant articles, this decision promises a better performance of the algorithm. The more articles are considered, the longer the process of calculating the recommendations takes. Our system currently uses only one news provider, but if the system grows, this could lead to a significant loss of performance. Our hybrid algorithm which incorporates collaborative filtering is also able to provide older articles if they are of high importance to the user (cf. Section 3.4).

The formula below computes the similarity between two articles (g and h),

$$sim(g,h) = \frac{\sum_{i \in W}(g_i * h_i)}{\sqrt{(\sum_{i \in W} g_i^2 * \sum_{i \in W} h_i^2)}} \quad , \qquad (1)$$

where

g,h *are* vectors representing articles with weighted keywords,

W *is* the set union of the particular keywords,

i *is* a keyword and

g_i,h_i *are* the weights of i in g and h, respectively.

In the computation of content-based similarity scores we only consider the relative dimension of the keyword weights. For the reason that user profiles have different dimensions compared to articles, the use of relative dimensions provides better results for our system. As an illustration of the main idea of the algorithm, let us consider the simple case where the user profile contains two keywords with the weights 5 and 10. Additionally there is another article with these two keywords but the weights are 1 and 2, respectively. In this

case the similarity is 1, because of the same relative dimensions of the article and the user profile.

The algorithm considers every article as an element in a vector space, where the keywords are forming the base. The coordinate of an article in the direction of a keyword is given by the weight of this keyword. If the keyword does not occur, the weight will be 0.

We normalize each article relative to the standard scalar product by dividing it by its absolute value. Consequently, the standard scalar product of the two normalized vectors conforms to the desired comparison features. Even if there are negative weights, e.g. for active suppressed keywords, the algorithm calculates similarities correctly.

In order to understand the similarity calculation better, we explain how the algorithm works for an article with itself (or another article with the same weight proportions). In this case, the scalar product is 1, because of the way the vectors are normalized. But if two articles have disjunctive keyword sets, the result is 0, because such vectors are orthogonal to each other.

In the end, the system sorts the articles by similarity descending and returns the 50 articles with the highest score.

3.3 Collaborative Filtering Component

In contrast to content-based filtering, a collaborative RS uses the ratings of other users to calculate the similarity of articles [1]. Different algorithms for item-based collaborative filtering exist. We explain some common algorithms in the following and explain our choice for a sports news RS. Therefore we refer to [12] and [14].

Vector-based / Cosine-based Similarity:

$$sim(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|i\| * \|j\|} \tag{2}$$

The first algorithm is the vector-based, also called cosine-based, similarity. In this algorithm, items are represented as two vectors that contain the user ratings. The similarity between item i and item j is calculated by the cosine of the angle between the two vectors. The "$\cdot$" denotes the dot product of vector $\vec{i}$ and vector $\vec{j}$ [12]. Due to the fact that cosine based similarity does not consider the average rating of an item, Pearson (correlation)-based similarity tries to solve this issue.

Pearson (Correlation)-based Similarity:

$$sim(i,j) = \frac{\sum_{u\in U}(R_{u,i}-\bar{R}_i)(R_{u,j}-\bar{R}_j)}{\sqrt{\sum_{u\in U}(R_{u,i}-\bar{R}_i)^2}\sqrt{\sum_{u\in U}(R_{u,j}-\bar{R}_j)^2}} \tag{3}$$

The first part of this algorithm is to find a set of users U that contains all users who rated both items i and j. These items are called co-rated items. Not co-rated items are not taken into consideration of this algorithm. This similarity calculation is based on how much the rating of a user deviates from the average rating of this item. $R_{u,i}$ represents the rating of a user u on item i and $\bar{R}_i$ denotes the average rating of an item i.

Adjusted Cosine Similarity:

$$sim(i,j) = \frac{\sum_{u\in U}(R_{u,i}-\bar{R}_u)(R_{u,j}-\bar{R}_u)}{\sqrt{\sum_{u\in U}(R_{u,i}-\bar{R}_u)^2}\sqrt{\sum_{u\in U}(R_{u,j}-\bar{R}_u)^2}} \tag{4}$$

Adjusted cosine similarity takes into account that the rating preferences of the different users differ. There are some user that always give low ratings, but on the other side there are users that rate highly in general. To avoid this drawback, this algorithm subtracts the average rating of a user $\bar{R}_u$ from each rating $R_{u,i}$ and $R_{u,j}$ on the items i and j.

The presented advantage is the reason why we apply the adjusted cosine similarity in our development. First, the system has to calculate the related articles list of all articles. To compute the related article list of an article, we iterate through the list of all articles. If the current article is not the same as the article to compare, we will calculate the similarity.

The function returns a value between minus one and one. Since the article rating range is from one to five, we map the similarity to the rating range by using the linear function:

$$sim = 2 * sim + 3 \tag{5}$$

There is one bigger problem in the adjusted cosine similarity calculation. When there is just one common user between articles, the similarity for those items is one, which is the highest value of the rating range. This is due to the subtraction of the average rating from the user's rating. To avoid the effect that the best rated articles are the articles with just one common user, we specified a minimum number of users that two articles need to have in common. In our implementation the minimum number of common users is five. When there are less than five common users, the articles are not considered in our related article list.

Afterwards, we sort the list by similarity. Moreover, we set a limit of 50 related articles to avoid additional expenses due to articles that are not considered for computation. When the related article list is calculated, we can predict the top articles for a user. For each article in the related articles list, we check if the user has already read the article. If that is the case, the article is not recommended anymore and the system jumps to the next article. If it is a new article, the prediction is calculated and added to the recommendation list. After calculating the prediction for every article, we sort the recommendation list by the predicted value.

The prediction $P_{u,i}$ can be calculated by the weighted sum method [12]:

$$P_{u,i} = \frac{\sum_{all\ similar\ items,N}(s_{i,N} * R_{u,N})}{\sum_{all\ similar\ items,N}(|s_{i,N}|)} \tag{6}$$

This approach is "computing the sum of the ratings given by the user on the items similar to i" [12]. Afterwards, each rating $R_{u,j}$ is weighted by the similarity between item i and item $j \in N$. The basic idea of this approach is to find items that are forecasted to be liked by the user. The top predicted items are recommended to the user.

A key advantage over content-based filtering techniques is the fact that collaborative RSs are able to provide a bigger variety of topics. Furthermore, with collaborative techniques, it is possible to provide event- or trend based recommendations, such as news about the World Cup. A pure

content-based RS is not be able to recommend news about the darts championship if the user has just read football articles before.

3.4 Weighted Hybrid Recommender

In this section, we explain how we combine the content-based and the collaborative components to a hybrid sports news RS.

As combination technique, we use the weighted hybrid strategy as described in [4]. For our first version, we decided to weight both components equally. The content-based component is important for recommending new articles even if no ratings exist. Additionally, the content-based system is able to provide content to users with special interests as well. Moreover, the content-based version is important, because of fan culture and constant interest in some topics. But we decided that the collaborative filtering part is as important as the content-based component, due to the event-based environment and the changing popularity of some sports. We want the system to be able to recommend articles that are attractive for just a small time slot. For example, many persons are interested in the Olympic Games, but not in the different kind of sports in general.

We determined that the weights are just applied if both components of our system recommend the corresponding article. Otherwise, additional requests have to be sent to calculate a combined score for each article. If just one component recommends the article, just the score of this component is taken with the full weight. Due to this procedure, we are able to provide recommendations of both components.

3.5 Implementation

We developed a dashboard widget which can be integrated in existing websites to provide personalized sports news recommendations. For the development of the front-end, we used the JavaScript framework AngularJS, the style sheet language Less and HTML5 local storage. Figure 1 shows a current screenshot of our recommender dashboard. Nine recommendations are presented at one time. When the mouse is moved over one article, the user can read it ("Ansehen") or reject the recommendation ("Entfernen").

It is critical to identify the user every time she or he accesses the RS to provide personalized recommendations. We avoided to implement a mandatory login as this could be a big obstacle for new users visiting a sports news website. Instead, we calculate a Globally Unique Identifier (GUID) which is then stored in HTML5 local storage without an expiration date. This is an important advantage for our RS. Due to the fact that HTML5 local storage has no explicit lifecycle, we can use it not only for user identification, but also for generating a profile of the user. Storing this data on client site is decreasing the amount of data stored on the server which makes the system more scalable. Only the item similarities and recommendations are calculated on the server, due to direct access to articles from our backend.

In order to get content-based recommendations, the client sends an Ajax request to a NodeJS server. Therefore, the user ID and the corresponding profile are sent as parameters. We decided to use an Ajax request due to the fact that the computation causes no overhead at site loading if it is done from JavaScript code. At our backend, the weighted keyword profile is sorted by the keyword name alphabetically. As mentioned, we receive articles published within the last three days. After obtaining those articles from our article repository, we add the suitable keywords to each of them. The weighted keywords are in the same form as the user profile to make them comparable to each other. The system calculates the similarity of the user profile with every article. Therefore, the union of the keyword sets is built. Subsequently, the similarity is computed using formula 1.

A JSON response sends the 50 articles with the highest score back to the client. The response is then processed by the Angular directive of the personalized dashboard. If the user removes an article, the next recommended article takes over its place. In addition, further statistics like the last read articles or last and next matches of the preferred team are displayed.

For the computation of collaborative recommended articles we use the same NodeJS server. In contrast to other systems, we do not store our data in a database. Due to the fact that we have to iterate through lists most of the time to compare ratings and users, we decided to use arrays to store our data within the application. The ratings provided by the user are collected in a rating variable that is kept in memory. It stores JSON objects with the user ID, the article path and the provided rating. Furthermore, the current date is used to distinguish current data from dated ratings that are not relevant for our system anymore.

To speed up the similarity computation, we adapt the average rating of a user every time providing a new rating. The average ratings are kept in an extra variable for performance reasons. The current average rating and the number of ratings provided by the given user is enough to adapt the average. Just a few basic arithmetic operations are necessary to avoid calculating the average from the rating variable every time from scratch. We minimize the accesses to the rating variable due to the fact that this variable is the main component of our server. Most of the requests read or write this variable. Every variable access that can be eliminated helps to improve the system's performance.

Moreover, we store a list of users as well as a list of articles to iterate through these arrays without generating them first. Using a list of all articles is primarily important when the system computes related articles. The list of related articles is updated every hour. A cronjob is executed every hour to consider current news as well. After one hour there are more ratings provided and the new item problem of a pure collaborative RS is suppressed.

For similarity calculation of two items, we need to find a set of users that contains all users who rated both items. Therefore, we generate a list of objects that contain the articles and all the users who rated the corresponding article. To compute the user set of two articles, we compare the two user lists and determine the intersection.

The combination of the content-based and the collaborative part of our RS is implemented in JavaScript. First, we send an Ajax request to our backend to collect the content-based recommended articles. In addition, another request is sent to our NodeJS server where the collaborative filtered articles are computed. If the collaborative filtered recommendations are returned correctly, the system computes the combination of both article sets. Finally, the recommended articles are returned and the JSON response is sent to the application.

In the news domain the age of an article is definitely one of the most important properties when the article's attrac-

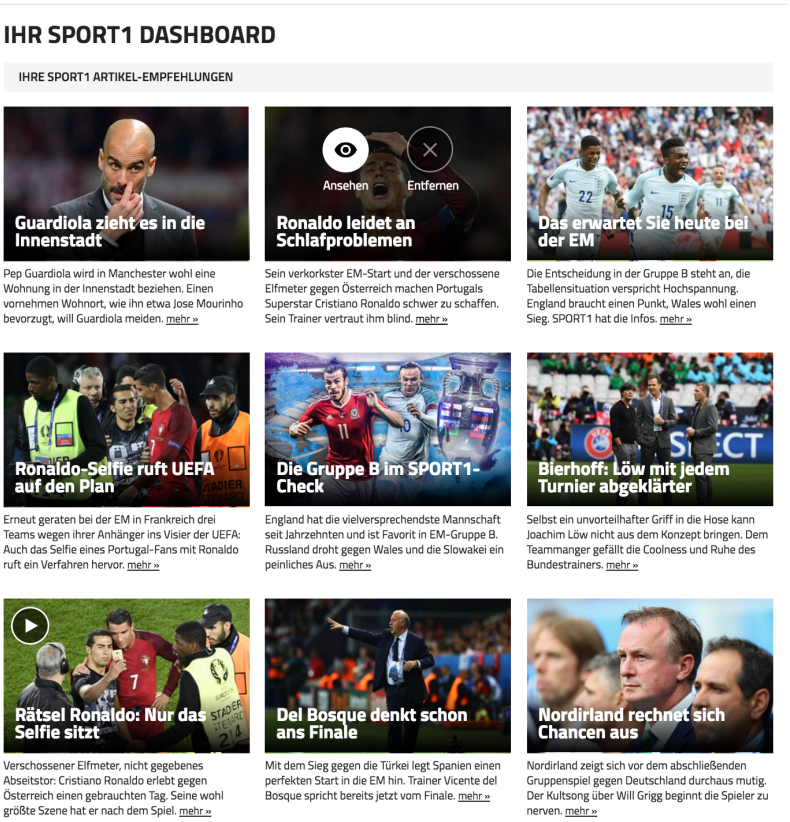


Figure 1: Recommender Dashboard

tiveness is determined for a user. Because of the recency problem, we decided to implement a route in our NodeJS server to remove dated ratings and articles from our system. Every two weeks a cronjob is executed and every rating that is older than four weeks is removed from the ratings table. The removal of those ratings implies the secondary effect that old users that do not exist anymore are removed as well. This is a very common scenario in our system, due to the fact that we identify the user by using HTML5 local storage. If the local storage is deleted, the old user ID does not occur anymore. We decided to use these time intervals, because our content-based version considers only articles published the last three days and we want to provide recommendations of articles older than a few days as well if an older article is getting popular again. In this case, our system is able to recommend those articles as well as long as ratings are provided in the last four weeks.

4. EVALUATION

We conducted user studies to evaluate our algorithms and the usability of our recommender dashboard. In this section we present the goals, the procedure and the results of our evaluation. We interpret our findings to answer the question how well content-based algorithms support user in receiving interesting sports news and if a hybrid algorithm can improve the performance of our RS.

4.1 Analysis of Usage Data of the Content-based Recommender

In order to collect usage data of real users, we tested the content-based approach on the live version of the Sport1 website. For this purpose, the recommender dashboard prototype is presented to one percent of the users. Due to the fact that the website is visited by thousands of users every day, one percent of the users is enough to evaluate not only the functionality but also the usability of our RS. In future, we will increase the amount of test users from time to time and adapt our implementation accordingly. We used Google Analytics to measure relevant Key Performance Indicators (KPIs) that help us to evaluate our solution. We analyzed how much the users clicked on the read and the remove button, respectively. Moreover, we tested how often the users navigated to articles they have already read by using the last read articles widget. In addition to the event tracking, we analyzed if there is an impact on the article ratings due to the new personalized dashboard. This is why we compare the average ratings of different articles. Articles are just taken into account, if they are rated by the one percent of users that can use the personalized dashboard.

At the end of our live study 5132 user IDs were registered on our server. This does not mean that more than 5000 different users used the dashboard due to the fact that every device has its own GUID and if the history of the browser is deleted, a new ID is generated. But there were enough users producing events we can track.

The click behaviors of the users give information about

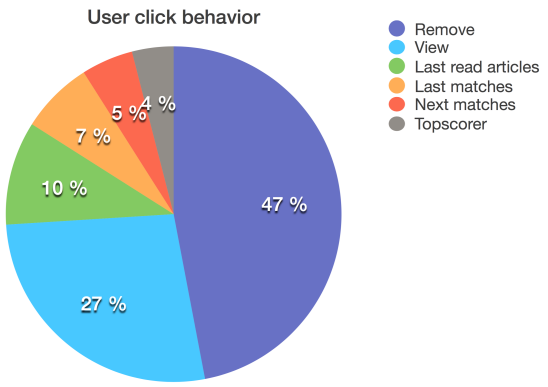


Figure 2: Analysis of the user's click behavior

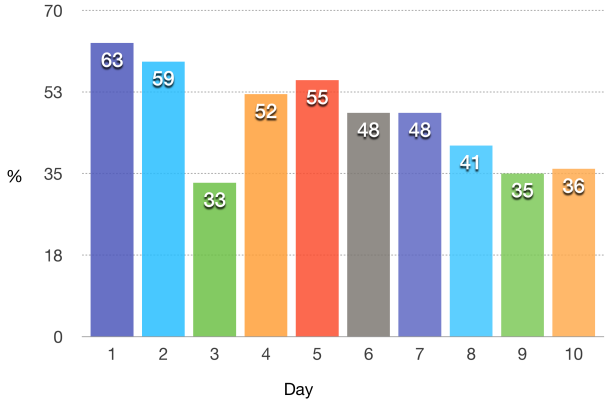


Figure 3: Daily clicks on remove (figures in per cent)

the user acceptance of the different components. Figure 2 illustrates how many clicks are executed on the different components of the personalized dashboard.

Almost 50 percent of the clicks were executed on the remove button of the news recommendation widget. On the first view, this number is quite high. But if we consider that at all the other buttons navigate the user to another page, it is obvious that the remove button is executed more often than all the other buttons. If the user clicks on remove, the article will disappear and a new one will be displayed in the dashboard. The user is then able to interact again with the dashboard. 27 percent of the clicks are executed on the view article button, which is a good proportion. Especially, if we consider that the RS is new, it is noticeable that after every third interaction, an article that potentially fits to the interests of the user, is recommended. To get better informations about the quality of the recommendations, we need to organize a long-term study. The sports news domain is very dynamic and the click behavior is changing depending on the current events. By that reason, the two week evaluation is not enough to ensure that the amount of clicks on an element is constantly similar. Around one quarter of all clicks are executed on links and buttons which are not part of the recommender dashboard but provide additional information such as last read articles and team-related statistics such as last and next matches and top scorers.

We expect that the quality of the recommendations increases with the time of use. To test this assumption, we analyzed the trend of the remove button clicks. Except for some days, the ratio of clicks on remove decreased with every day performing our testing (cf. Figure 3). The exceptions may base on new users or users that do not read many articles on the website. If none or just a few articles are read before using our dashboard, the quality of the recommendations will be low. Since the dashboard is just presented to one percent of the users, we are not able to give evidence that the subjective quality will be the same when publishing the dashboard to all users. With increasing the number of testers, the ratio of remove button clicks increases at the beginning and then falls again with the time of use. We detected this when we released the dashboard on the website.

To analyze if the dashboard has an effect on the article ratings, we compared three types of articles. First, articles that are bad rated in general, second average rated articles and finally articles that are high rated in general. A generally bad rated article gets a better score from our RS users. This is mainly due to the fact that we use the implicit feedback of three stars if a user reads the article. Moreover, the bulk of the users is not providing any rating for an article. So the average rating of the testers is almost at three stars for bad and average rated articles. For high rated articles, the RS users scored a little lower in general. The chance to get such an article provided by our system is higher due to the fact that more comparable users are available for the most read articles. If the user clicks on remove, the lowest rating of one star is implicitly provided and the average rating is decreasing. Since the personalized dashboard is not established on the website, we can sum up that the recommendations have almost no effect on the rating scores. This may change if the users will use the dashboard as their first contact point on the website.

It was also noticeable that the users want to read already read articles again. The last read article widget helps them to navigate back and easily get an overview of the last interactions. We expect that the amount of clicks in this widget will decrease in the future. Since new articles are potentially more attractive for a user, we can not imagine that every tenth click is executed on an already read article. We believe that the users in the study were curious and wanted to test this new feature.

4.2 Comparison of Content-based and Hybrid Recommendations

Our hybrid algorithm extends the presented content-based approach by a collaborative filtering component. This algorithm is not part of the live version of the RS yet. We tested the hybrid recommendations with a selected user group. We paid attention to choose persons from different backgrounds and with diverging interests to ensure comparability to our users.

The participants had to rate the RS in two scales on a scale from 1 (worst rating) to 5 (best rating): How well the recommendations fit their interests and how diversified they are. The pure content-based solutions served as a baseline algorithm. In total, we received 40 completed questionnaires for the content-based approach and 20 for the extended, hybrid RS.

The results show that the recommendations are not diver-

sified enough in our pure content-based approach (Ø2.9), but they improve in our hybrid implementation where the average rating was 3.3. The content-based recommendations are representing the interests of the user (Ø3.4) which shows us that the dashboard provides additional value. This value did not change in our hybrid version. It was noticeable that the more frequent a user is visiting the website, the more he is satisfied with the result of the recommendations. The users that visited the website every day gave an average rating of 3.6. This confirms that the quality of our recommendations increases over time.

4.3 Usability of the Recommender Dashboard

Besides evaluating our recommendation algorithms, we asked the study participants to rate the usability of our recommender dashboard. We used the well-established System Usability Scale (SUS) [3]. This questionnaire consists of ten questions providing a global view of subjective assessments of usability. Participants respond using a Likert scale with five response options; from Strongly agree to Strongly disagree. Furthermore, our participants were allowed to add further thoughts in a free-text field.

To calculate the SUS score, the answers for each question are converted to a new score from 0 to 4 where 4 is the best score and 0 is representing the worst possible answer of this question. Afterwards the different scores are added together and multiplied by 2.5 to get a ranking value between 0 and 100 [3]. Every SUS score above 68 is considered as above average, everything lower than 68 as below average [13]. The average scores of each question, collected in our user feedback, are shown in Table 1.

The score is calculated by adding the scores and multiplying the sum with 2.5:

$$score = sum * 2.5 = 34.8 * 2.5 = 87 \quad (7)$$

The score of 87 exceeded our expectations although we attached great importance to the design and usability of our system. This was required because the dashboard is implemented on the live website of Sport1. Nevertheless the users mentioned some desires concerning the usability. For example, some users wished to change the design by choosing their own colors. Since these informations have not an direct impact on our RS implementation, we will not deepen these suggestions here.

4.4 Discussion

The user study results show that our content-based RS is a promising approach to suggest sports news to users. The recommendations fit the users' interests and improve when the users provide more feedback. Nevertheless, the diversity of the recommended articles remains to be low. This is a typical problem of pure content-based RS and can be overcome by using a hybrid solution. We extended the RS by a collaborative filtering component which increased the diversity of the recommendations.

As described before, it is very important that a news RS provides current articles. Especially in sports, the environment is very dynamic and the news topics are changing all the time. For that reason the system can not be a pure collaborative RS. With collaborative filtering it is almost impossible to recommend new items. But this problem can be solved by using a content-based component as well. Content-based RSs can provide content that fits to the general interests of the users. In addition, attention should be paid to event based interests, e.g. the Super Bowl, an event that is closely followed by many people. If a user has no interests in American Football in general, the content-based RS does not provide articles about the Super Bowl. So there must be a combination of both techniques to benefit from the strengths of each component.

5. CONCLUSION AND FUTURE WORK

In this work, we tackled the problem of recommending sports news. Sports news are a special case in the field of news recommendations as users often come with a strong emotional attachment to selected sports, teams or players. Furthermore, the interest in a topic is event-driven and can suddenly change. We developed a content-based RS that creates user profiles based on implicit feedback the user shares when reading articles. Using automatically created keywords, the similarity between articles can be measured and the relevance for the user can be predicted. This approach delivers accurate recommendations but lacks diversity. In a first prototype, we designed and evaluated a hybrid algorithm that extends our content-based RS by a collaborative component. This hybrid approach increases diversity and also allows to recommend older articles if they are of particular interest for the user.

To improve the quality of the hybrid recommendations, we will adjust our implementation from time to time and test if the adaptions serve their purpose. First, we will test different weights for the two components. One idea is to increase the weight of the content-based version. The decrease of the weight of the collaborative version does not exclude the event-based recommendations. Even if the collaborative part does just count one third, it is able to provide recommendations because if the article is only recommended by our collaborative version, just the score of this component is taken into account. If both components provide this article recommendation, the content-based version is more adapted to the users interests. To find out which weight ratio is the best for our case, we have to analyze the implicit and explicit user feedback for a longer time period. The evaluation of the weights is just meaningful if the feedback is collected for a few months to avoid temporally fluctuation, which is quite common in the news domain.

Furthermore, we want to implement a switching hybrid as well. If there is a new item, the collaborative filtering method can not provide recommendations from the first second. This is the strength of our content-based version. The RS has to switch to the content-based version if the article is newer than a specific date. Recommendations for a new user are calculated by our collaborative filtering component to handle new users as well as the preferences at the first use of the system are not sufficient to compute pure content-based recommendations. If a larger user profile is constructed and an article is not published in the last minutes, the combination of both techniques will be applied as described before.

We tested our first developments in a two-week user study. Our content-based RS has been tested with live users while the hybrid approach was only accessible for a selected user group. In future, we want to conduct larger studies with more users for all algorithms we develop. Our first results will serve as the baseline for future extensions and other algorithms.

Table 1: Questions and Results of the SUS questionnaire

Number	Question	Average Score
1	I think that I would like to use this system frequently.	3.4
2	I found the system unnecessarily complex.	3.3
3	I thought the system was easy to use.	3.6
4	I think that I would need the support of a technical person to be able to use this system.	3.9
5	I found the various functions in this system were well integrated.	3.3
6	I thought there was too much inconsistency in this system.	3.1
7	I would imagine that most people would learn to use this system very quickly.	3.6
8	I found the system very cumbersome to use.	3.6
9	I felt very confident using the system.	3.2
10	I needed to learn a lot of things before I could get going with this system.	3.8

6. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.

[2] Y. A. Asikin and W. Wörndl. Stories around you: Location-based serendipitous recommendation of news articles. In *Proceedings of 2nd International Workshop on News Recommendation and Analytics*, 2014.

[3] J. Brooke. SUS-A quick and dirty usability scale. *Usability evaluation in industry*, pages 189–194, 1996.

[4] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, Nov. 2002.

[5] I. Cantador, A. Bellogín, and P. Castells. News@hand: A semantic web approach to recommending news. In W. Nejdl, J. Kay, P. Pu, and E. Herder, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems: 5th International Conference, AH 2008, Hannover, Germany, July 29 - August 1, 2008. Proceedings*, pages 279–283. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[6] I. Cantador, A. Bellogín, and P. Castells. Ontology-based personalised and context-aware recommendations of news items. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '08, pages 562–565, Washington, DC, USA, 2008. IEEE Computer Society.

[7] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems*, volume 60. Citeseer, 1999.

[8] T. De Pessemier, S. Leroux, K. Vanhecke, and L. Martens. Combining collaborative filtering and search engine into hybrid news recommendations. In *Proceedings of the 3rd International Workshop on News Recommendation and Analytics (INRA 2015) co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 20, 2015.*, pages 14–19, 2015.

[9] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, Mar. 1997.

[10] J. Liu, P. Dolan, and E. R. Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*, 2010.

[11] Ö. Özgöbek, J. A. Gulla, and R. C. Erdur. A survey on challenges and methods in news recommendation. In *Proceedings of the 10th International Conference on Web Information Systems and Technologies*, pages 278–285, 2014.

[12] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[13] J. Sauro. Measuring Usability with the System Usability Scale (SUS), 2011. Retrieved June 20, 2016 from http://www.measuringu.com/sus.php.

[14] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Hindawi Publishing Corporation*, 2009, 2009.

[15] A.-H. Tan and C. Teo. Learning user profiles for personalized information dissemination. In *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on*, volume 1, pages 183–188, May 1998.

News Article Position Recommendation Based on The Analysis of Article's Content - Time Matters

Parisa Lak
Data Science Laboratory
Ryerson University
Toronto, Canada
parisa.lak@ryerson.ca

Ceni Babaoglu
Data Science Laboratory
Ryerson University
Toronto, Canada
cenibabaoglu@ryerson.ca

Ayse Basar Bener
Data Science Laboratory
Ryerson University
Toronto, Canada
ayse.bener@ryerson.ca

Pawel Pralat
Data Science Laboratory
Ryerson University
Toronto, Canada
pralat@ryerson.ca

ABSTRACT

As more people prefer to read news on-line, the newspapers are focusing on personalized news presentation. In this study, we investigate the prediction of article's position based on the analysis of article's content using different text analytics methods. The evaluation is performed in 4 main scenarios using articles from different time frames. The result of the analysis shows that the article's freshness plays an important role in the prediction of a new article's position. Also, the results from this work provides insight on how to find an optimised solution to automate the process of assigning new article the right position. We believe that these insights may further be used in developing content based news recommender algorithms.

CCS Concepts

•**Information systems** → *Content ranking; Recommender systems;*

Keywords

Content-based Recommender System; Text Analytics; Ranking models; Time-based Analysis

1. INTRODUCTION

Since 1990s the Internet has transformed our personal and business lives and one example of such a transformation is the creation of virtual communities [2]. However, there are challenges in the production, distribution and consumption of this media content [8]. Nicholas Negroponte has contented that moving towards being digital will affect the economic model for news selection and the users' interest play a bigger role for news selection [7]. Therefore, users actively participate in online personalized communities and they expect the online news agency to provide as much personalized services as possible. Such demand, on the other hand, puts pressure on the news agency to employ the most recent technology to satisfy their users.

Our research partner, the news agency, is moving towards providing a more personalized service to their subscribed users. Currently, the editors make the decision on which article to be placed in which section and to whom the article should be offered (i.e. the subscription type). This decision is purely made based on their experience. Similarly, the position of the news within the first page of the section is decided by the editors. The company would like to first automate the decision on article position process and in the second step to provide personalized recommendations to their users. They would like to position the news on each page based on the historical behavior of each user available through the analysis of user interaction logs.

In this work, we investigate different solutions to optimize and automate the process of positioning the new articles. The results of this study may further be used towards building personalized news recommendation algorithms for subscribed users at different tiers. The high level research question that we address in this study is:

RQ- How to predict an article's position in a news website?

To address this question, we evaluate three key factors. First, we compare three text analytics techniques to find the best strategy to analyze the content of the available news articles. Second, we evaluate different classification techniques to find the best performing algorithm for article position prediction. Third, we investigate the impact of the time variable on the prediction accuracy. The main contribution of this work is to provide insights to researchers and practitioners on how to tackle a similar problem by providing the results from a large scale real life data analysis.

The rest of this manuscript is organized as follows: Sec-

CBRecSys 2016, September 16, 2016, Boston, MA, USA.

tion 2 provides a summary of prior work in this area. Section 3 describes the data and specifies the details of the analysis performed in this work. The results of the analysis are provided in Section 4 that is followed by the discussion and future direction in Section 5.

2. BACKGROUND

To automate the process of assigning the right position to a news article, researchers provide different solutions. In most of the previous studies, a new article's content is analyzed using text analytics solutions. The result of the analysis is then compared with the analysis of previously published articles. The popularity of the current article is predicted based on the similarity of this article with the previously published articles. Popularity is considered with different measures throughout literature. For example, Tatar et al. predicted the popularity of the articles based on the analysis of the comments provided by the users [10]. Another study, evaluated the article's popularity based on the amount of attention received by counting the number of visits [5]. Another article popularity measure that was used in a recent work by Bansal et al. is based on the analysis of comment-worthy articles. Comment-worthyness is measured by the number of comments on a similar article [1].

In the current work, we considered the popularity measure to be a combination of some of the aforementioned measures. Specifically, we used measures such as article's number of visits, duration of visit, the number of comments and influence of article's author to evaluate the previous article's popularity. The popularity measure is then used towards the prediction of article's position on the news website.

To evaluate the content of the article and find the relevant article topics several text analytics techniques has been used by different scholars [4]. Among all, we selected three commonly used approaches in this study. The three approaches are Keyword popularity, TF-IDF and Word2Vec that will be explained in section 3.

3. METHODOLOGY

In this section we specify the details of our data and we outline the details of the methodology used to perform our analysis. The general methodology framework that was used in this study is illustrated in Figure 1.

3.1 Data

One year historical data was collected from the news agency's archive. Information regarding the articles published from May 2014 to May 2015 was extracted from the agency's cloud space. One dataset with the information regarding the content of the articles as well as its author and its publication date and time was extracted trough this process. This dataset is then used to generate the keyword vector.

As illustrated in Figure 1, another dataset was also extracted from the news agency's data warehouse. The information regarding the popularity of the article, such as Author's reputation, Article's freshness and Article type were included in this dataset. The dataset also contained the news URL as article related information. This piece of information provides the details regarding the article's section and article's subscription type. The current position of the article is also available in the second dataset. The popularity of the article is then calculated based on available features

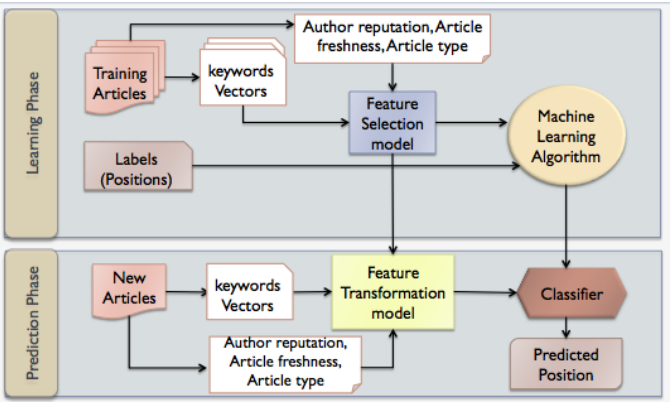


Figure 1: Step by step Prediction Methodology

and the position of the article in the website. This information along with the information from keyword vectors are then used as an input to the machine learning algorithms.

3.2 Analysis

We first analysed the content of each article available in the first dataset, using three text analytics techniques. Keyword Popularity, TF-IDF and word2vec were used to perform these set of analyses.

For the Keyword Popularity technique, we extracted the embedded keywords in the article's content and generated keyword weights based on the combination of two factors: the number of visits for a particular keyword and the duration of the keyword on the website. For instance, if the article had a keyword such as "Canada", we evaluated the popularity of "Canada" based on the number of times it occurred in the selected section and the number of times an article with the keyword "Canada" was visited previously.

In TF-IDF technique, TF measures the frequency of a keyword's occurrence in a document and IDF refers to computing the importance of that keyword. The output from this technique is a document-term matrix with the list of the most important words along with their respective weight that describe the content of a document [9]. We used nltk package in python to perform this analysis over the content of each article.

The last text analytics technique used in this study is word2vec. This technique was published by Google in 2013. It is a two-layered neural networks that processes text[6].This tool takes a text document as the input and produces a keyword vector representation as an output. The system constructs vocabulary from the training text as well as numerical representation of words. It then measures the cosine similarity of words and group similar words together. In another words, this model provides a simple way to find the words with similar contextual meanings [9].

A set of exploratory data analysis was performed on the second dataset to find the most relevant features to define article's popularity. Based on the result from this set of analysis we removed the highly correlated features. The popularity measure along with the position and the keyword vector of each article is then used in 4 main classification algorithms: support vector machine (SVM), Random forest, k-nearest neighbors (KNN) and Logistic regression [3]. The result of the analysis are only reported for the first two al-

gorithms (i.e. SVM and Random Forest) as they were the best performing algorithms among the four for our dataset.

The steps to perform the prediction analysis also illustrated in Figure1. As shown, the analysis is mainly performed in two phases denoted as "Learning phase" and "Prediction phase". In the learning phase the training dataset is cleaned and preprocessed and the features to be used for the evaluation of popularity are selected based on the exploratory analysis. All observations (i.e. articles) in this dataset are also labeled with their current positions. In the prediction phase, the article content is analyzed and the keyword vectors are created based on the three text analytics techniques. Then, the popularity of the article is calculated based on available features. The test dataset is then passed through the classifier, which predicts the position of the article. The accuracy of prediction is evaluated based on the number of correctly classified instances to the total number of observations and can be computed with Equation 1.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \qquad (1)$$

The result of the analysis is reported in the following section.

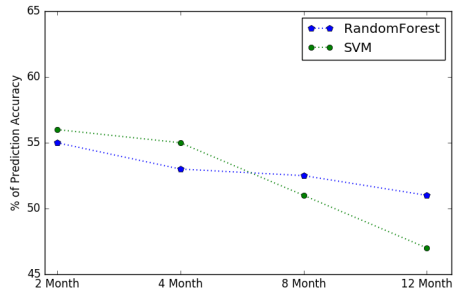
4. RESULTS

The set of graphs in Figure 2 illustrates the percentage of prediction accuracy trend for articles' positions in 4 different scenarios using the two classification algorithms. The blue graph shows the accuracy trend for RandomForest classification algorithm, while the green graph reports the accuracy for the SVM. The 4 scenarios are based on the training data used in the machine learning algorithms. The first points from the left shows the accuracy for the scenario, when the training set contains the articles from 2 months prior to the publication of the test article. Similarly, the second point from the left shows the scenario in which the training set contains articles from 4 months prior to the publication of the test article and so on for the 8 months and 12 months scenario.

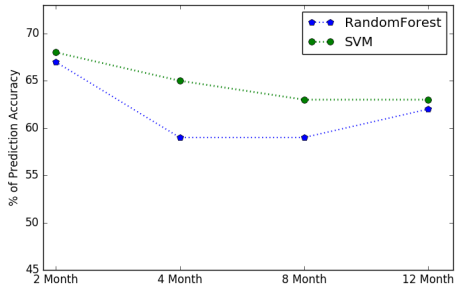
Figure 2(a) shows the accuracy results for the articles when their content (for both training and test dataset) is analyzed based on Keyword Popularity technique. In this graph we observe that the accuracy of the prediction algorithm is related to the time frame factor used to build the training set. More specifically, both algorithms perform best while the most recent articles are used in the training set. It clearly shows that the performance of both SVM and Random Forest is dependant on the time frame that is used to define the training set.

Figure 2(b) provides the accuracy for the analysis of the prediction in the case when the articles are analyzed by TF-IDF technique. The result of the analysis for this content analysis technique further confirms that the accuracy of prediction is dependant on the time frame selected to define the training set. For this type of article content analysis, SVM always works superior to RandomForest in terms of accuracy.

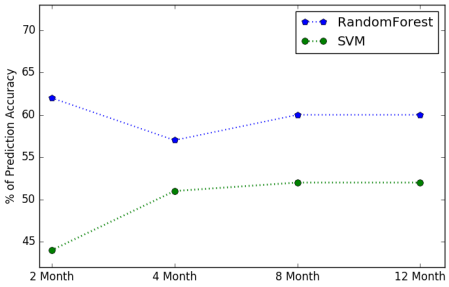
Figure 2(c) shows the result of the prediction for the articles that are evaluated by Word2Vec technique. The result from this graph is different from the previous two graphs. The accuracy for the most recent articles using SVM shows to be lower from other scenarios, however the difference between the accuracy of the other time dependent scenarios are not shown to be large. Although, SVM shows a different accuracy trend for this text analytics technique, the accuracy results for the Random Forest algorithm seems to be consistent with the results from prior analysis. Specifically, while using Woed2Vec and Random Forest algorithm, the best performance is gained through the use of the most recent articles in the training set. On the contrary, the result for this text analytics technique and the use of SVM algorithm works best, while using the older articles. Nevertheless, SVM is not considered as the best performing algorithm for this text analytics technique.



(a) Keyword popularity



(b) TF-IDF



(c) Word2Vec

Figure 2: Prediction accuracy for SVM and Random forest in 4 time frame scenarios (2, 4, 8 and 12 months) using different article content analysis techniques

To better illustrate the performance of each text analytics techniques based on the time dependent scenarios Figure 3 is provided.

Figure 3 shows the result from the best performing algorithm for the three content analytics techniques within the 4 time dependent scenarios. The blue graph shows the perfor-

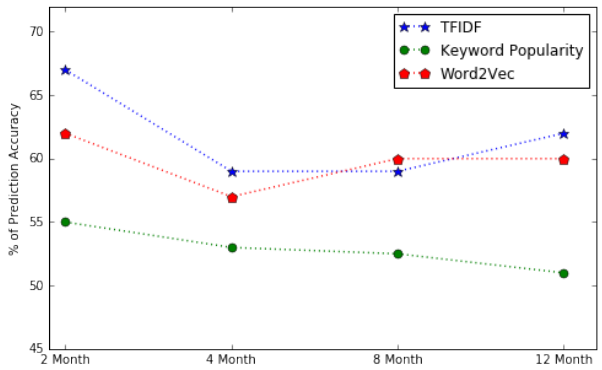


Figure 3: Prediction accuracy based on the three content analysis techniques for the 4 time frame scenarios

mance of SVM for TF-IDF technique and the green graph and the red graph show the accuracy result for Random Forest for Keyword popularity and Word2Vec, respectively. This figure shows that for all the three content analysis techniques, the best prediction performance is achieved while the fresh articles are used for training purposes. The accuracy is always dropped as old articles are added to the training set in the 4 month scenario. In Word2vec technique, the accuracy increases when the 8 month prior articles are used for training. However, still the best performance is attained while using more recent documents.

Another observation from this analysis is that TF-IDF technique provides the best text evaluation that further generates higher prediction accuracy for article's position.

5. DISCUSSION & FUTURE DIRECTION

Personalized news recommendation is a recently emerged topic of study based on the introduction of the interactive online news media. The decision on the news presentation is made based on the assigned position of the article within the news website. The position of the article can be assigned based on the popularity of the article. The popularity of the article can be predicted based on the analysis of its content and the similarity of the article's content to the previously published articles. Previous article's popularity is measured based on different popularity measures. In this study, we used a combination of article's popularity measure attributes as well as the attributes from the analysis of the articles' content to predict the position of a new article.

We evaluated the impact of the three key factors on the prediction of new article's position. The results from the analyses provide evidence that all three factors under investigation in this study plays a role in the accuracy of prediction. One of the important findings from this work is that the result of the analysis of a new articles content should only be compared with the recent articles. The analysis shows that as the older articles are used as an input to the prediction algorithm the accuracy of the system drops in almost all cases. Also, the best performing prediction algorithm shows to be dependent on the text analytics techniques used in the analysis of the article's content. Regardless of the prediction algorithm the best text analytics technique for the current dataset is shown to be TF-IDF.

The results from this study can cautiously be extended to other datasets. To avoid the impact of sampling biases we used 10 fold cross validation technique for our prediction models. Also, the analysis of the large scale real life data minimizes this threat to the validity of the result of this study. In our future work, we will use the the results from this study as well as the features detected through the exploratory analysis to design a personalized news recommendation system.

6. ACKNOWLEDGMENTS

The authors would like to thank Bora Caglayan, Zeinab Noorian, Fatemeh Firouzi and Sami Rodrigue who worked at different stages of this project. This research is supported in part by Ontario Centres of Excellence (OCE) TalentEdge Fellowship Project (TFP)-22085.

7. REFERENCES

[1] T. Bansal, M. Das, and C. Bhattacharyya. Content driven user profiling for comment-worthy recommendations of news and blog articles. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 195–202. ACM, 2015.

[2] P. J. Boczkowski. *Digitizing the news: Innovation in online newspapers*. mit Press, 2005.

[3] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[4] S. Lee and H.-j. Kim. News keyword extraction for topic tracking. In *Networked Computing and Advanced Information Management, 2008. NCM'08. Fourth International Conference on*, volume 2, pages 554–559. IEEE, 2008.

[5] L. Li, D.-D. Wang, S.-Z. Zhu, and T. Li. Personalized news recommendation: a review and an experimental investigation. *Journal of Computer Science and Technology*, 26(5):754–766, 2011.

[6] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[7] N. Negroponte. *Being digital*. Vintage, 1996.

[8] J. V. Pavlik. *Journalism and new media*. Columbia University Press, 2001.

[9] N. Pentreath. *Machine Learning with Spark*. Packt Publishing Ltd, 2015.

[10] A. Tatar, J. Leguay, P. Antoniadis, A. Limbourg, M. D. de Amorim, and S. Fdida. Predicting the popularity of online articles based on user comments. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, page 67. ACM, 2011.

Using Visual Features and Latent Factors for Movie Recommendation

Yashar Deldjoo
Politecnico di Milano
Via Ponzio 34/5
20133 Milan, Italy
yashar.deldjoo@polimi.it

Mehdi Elahi
Politecnico di Milano
Via Ponzio 34/5
20133 Milan, Italy
mehdi.elahi@polimi.it

Paolo Cremonesi
Politecnico di Milano
Via Ponzio 34/5
20133 Milan, Italy
paolo.cremonesi@polimi.it

ABSTRACT

Item features play an important role in movie recommender systems, where recommendations can be generated by using explicit or implicit preferences of users on attributes such as genres. Traditionally, movie features are human-generated, either editorially or by leveraging the wisdom of the crowd.

In this short paper, we present a recommender system for movies based of Factorization Machines that makes use of the *low-level visual* features extracted automatically from movies as side information. Low-level visual features – such as lighting, colors and motion – represent the design aspects of a movie and characterize its aesthetic and style.

Our experiments on a dataset of more than 13K movies show that recommendations based on low-level visual features provides almost 10 times better accuracy in comparison to genre based recommendations, in terms of various evaluation metrics.

1. INTRODUCTION

Video-on-demand applications are characterized by the large amount of new video content produced every day. As an example, hundreds of hours of video are uploaded to YouTube every minute.

Recommender Systems based solely on Collaborative Filtering (CF) fail to provide reliable recommendations, as the large number of newly produced movies have no or very few ratings. Side information about movies (e.g., genre, cast) can be exploited to help CF deal with the new-item problem [21]

A necessary prerequisites for CF with side-information is the availability of a rich set of high-level descriptive *attributes* about movies. In many cases, such information is human-generated and prone to biases or errors.

In contrast to human-generated attributes, the content of movie streams is itself a rich source of information about low-level stylistic features that can be used to provide movie recommendations. Indeed, by analyzing a movie stream content and extracting a set of informative features, a recommender system can make personalized recommendations tailored to the users' tastes. This is particularly beneficial in the new item scenario, i.e., when a new video is added to the catalogue with absolutely no attributes available [19, 12]. While this is an interesting research direction, it has received only marginal attention of the researchers in the field.

CBRecSys 2016, September 16, 2016, Boston, MA, USA.

In this paper, we show how to use low-level visual features extracted automatically from video files as input to the recommendation algorithm. We have identified a number of visual features that have shown to be very representative of the users' feelings, according to *Applied Media Aesthetics* [23]. Our features are part of the low-level visual of a movie, and are indicative of lighting, colors and motion in the movies [9].

We have performed an exhaustive evaluation by comparing the low-level visual features, w.r.t., a more traditional set of features, i.e., genre. We have used one of the state-of-the-art recommendation algorithm, i.e., Factorization Machines (FM) [18], and fed it with either set of movie features. We have computed different relevance metrics (precision, recall, and F1) over a large dataset of more than 13M ratings provided by 182K users to more than 13K movie trailers. We have used trailers (instead of full-length movies) in order to have a scalable recommender system. In early works, we have shown that low-level features extracted from trailers of movies are equivalent to the low-level features extracted from full-length movies, both in terms of feature vectors and quality of recommendations [10, 11]. We have also performed discriminative analysis, using both trailers and full-length movies, in order to better understand the effectiveness of each low-level visual feature, individually and in combination with the others, on the performance of the recommender system. The analysis have shown high similarity between the low-level features extracted from trailers and full-length movies [10].

The results of this paper shows that recommendations based on low-level visual features achieve an accuracy, almost 10 times better than the accuracy of genre-based recommendations.

This paper extends our previous work [10], where we presented some preliminary results obtained on a much smaller dataset of 160 movies, and simpler recommendation algorithm (a content-based algorithm based on cosine similarity between items).

Our work provides a number of contributions to the research area of movie recommendation:

- we propose a novel RS that automatically analyzes the content of videos and extracts a set of low-level visual features, and uses them as side information fed to Factorization Machines, in order to generate personalized recommendations for users
- we evaluate the proposed RS using a dataset of more than 13K movies, from which we extracted the low-

level visual features

- the dataset, together with the user ratings and the visual features extracted from the movies, is available for download[1].

2. RELATED WORK

Multimedia recommender systems typically exploit *high-level* features in order to generate movie recommendation [5, 15, 6, 16, 7]. This type of features express semantic properties of media content that are obtained from structured sources of meta-information such as databases, lexicons and ontologies, or from less structured data such as reviews, news articles, item descriptions and social tags.

In contrast, in this paper, we propose exploiting *low-level* features to be exploited for recommendation generation. Such features express stylistic properties of the media content and they are extracted directly from multimedia content files [10]. This approach has been already investigated in music recommendation domain [2, 20]. However, it has received marginal attention in movie recommendation domain.

The very few approaches in the video recommendation domain which exploit low-level features only consider scenarios where low-level features are used jointly with high-level features to improve the quality of recommendations. The work in [22] proposes a video recommender system, called *VideoReach*, which incorporate a combination of high-level and low-level video features (such as textual, visual and aural) in order to improve the click-through-rate metric. The work in [24] proposes a multi-task learning algorithm to integrate multiple ranking lists, generated by using different sources of data, including visual content.

This paper addresses a different scenario [12], i.e., when the high-level features are not available (e.g., in the new item scenario). Accordingly, the proposed recommender system can analyze the movies, extract a set of low-level visual features, and use it effectively to generate personalized recommendations.

3. METHOD DESCRIPTION

Video content features can be roughly classified into two hierarchical levels:

High-level (HL): the semantic features that deal with the concepts and events in a movie, e.g. the plot of a movie which consists of a sequence of events.

Low-level (LL): the stylistic features that define the mise-en-scene characteristics of the movie, i.e., the design aspects that characterize aesthetic and style of a movie.

Recommender systems in the movie domain use HL features, usually provided by a group of domain experts or by a large community of users, such as movie genres (structured features, high level). Our focus in this work is mainly on the LL visual features. The influence of these elements in the perception of a movie in the eyes of a viewer has been observed in the works by [4, 13] and were identified in our previous works [10, 11, 9]. The method used to extract low-level visual features and to embed them in movie recommendations is composed of the following steps as shown in Figure 1: (i) Video structure analysis, (ii)Video content analysis, and (iii) Recommendation.

[1]http://recsys.deib.polimi.it/

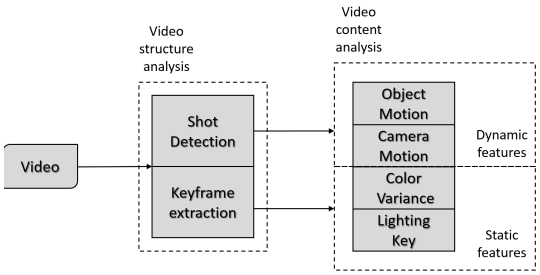


Figure 1: Generic framework of our video analysis system

Video structure analysis aims at segmenting a video into a number of structural elements including shots and key frames. Shots are sequences of consecutive frames captured without interruption by a single camera. From each shot, we extract the middle frame as the representative key frame, inspired by [17]. Two classes of features are extracted in the next stage, i.e., the video content analysis stage: dynamic features capturing the temporal aspects in a video (e.g. object motion and camera motion) are calculated from consecutive frame within each shot and static feature capturing spatial aspect (e.g. color variance and lighting key) are extracted from key frames. The visual feature vector $\mathbf{f_v}$ is composed of the following elements:

$$\mathbf{f_v} = (\overline{L}_{sh}, \mu_{cv}, \sigma^2_{cv}, \mu_{\overline{m}}, \mu_{\sigma^2_m}, \mu_{lk}, n_s) \quad (1)$$

where $\overline{L}_{sh}$ is the average shot length, μ_{cv} and σ^2_{cv} are the mean and the standard deviation of color variance across key frames, $\mu_{\overline{m}}$ and $\mu_{\sigma^2_m}$ are the mean of motion average and the mean of motion standard deviation across all frames, μ_{lk} is the mean lighting key over key frames and n_s is the number of shots.

3.1 Recommendation algorithm

In order to generate recommendations using our low-level visual features, we adopted a complex algorithm called Factorization Machines (FM) [18]. FM is one of the most advanced predictors and it is a combination of well-known Support Vector Machines (SVM) with factorization models. FM is a generic predictor that can work with any feature vector such as our visual features, or genre. It has been already tested and has shown excellent performance in content-based recommendation [14, 8], and high scalability in big datasets [18]. FM computes rating predictions as a weighted combination of the latent factors, low-level visual features, and biases. FM models complicated relationships in the data.

We have used two baselines: genre-based FM, which uses the item feature vector of length 19 (i.e., the number of unique genres), and top-rated non-personalized recommender.

3.2 Normalization

After the extraction of the low-level visual features, they have been normalized adopting 3 types of normalization:

Logarithmic: for every low-level feature (out of 7), the values of that feature is passed through a logarithmic function (natural logarithm). This changed the distributions to be approximately normal, as the original features in the dataset had a distribution similar to log normal distribution.

Quantile: for every low-level feature, the values of that

feature are normalized by applying quantile normalization [3]. This would change the distribution of all the features to be similar.

Log-Quantile: for every low-level feature, the values of that feature are normalized by applying logarithmic normalization (natural logarithm). Then, quantile normalization is applied to make the distribution of all the features to be similar.

Finally, regardless of the normalization type, we scaled the values of all features to the range of 0-1.

4. RESULTS

We have used the latest version of the Movielens dataset which contains 22,884,377 ratings provided by 247,753 users to 34,208 movies (sparsity 99.72%) [1]. For every movie, we queried Youtube and downloaded the trailer, if available. The final dataset contains **13M** ratings provided by **182K users** to **13,373** movies classified along 19 genres: *Action*, *Adventure*, *Animation*, *Children's Comedy*, *Crime*, *Documentary*, *Drama*, *Fantasy*, *Film-Noir*, *Horror*, *Musical*, *Mystery*, *Romance*, *Sci-Fi*, *Thriller*, *War*, *Western*, and *unknown*. Low-level features have been automatically extracted from trailers. We have used trailers and not full videos in order to have a scalable recommender system.

4.1 Experiment 1: Normalization

In order to evaluate the effectiveness of different normalization techniques for low-level features, we have fed the FM model with the low-level visual features. Table 1 presents the results of the evaluation. Different normalizations of the data result in different performances of low level features. Indeed, our observation shows that the best normalization is log-quantile, which by far, outperforms the other two methods, in terms of all evaluation metrics, we considered. The other two methods, have shown similar performance with no significant differences.

These results may point out that the main difference between the best method and the other two is the adoption of both logarithmic and quantile normalizations [3]. Indeed, this may indicate that both logarithmic and quantile normalizations are very necessary to be adopted to obtain the visual low-level feature values that can well represent the movie trailers, and at the same time, distinguish them from each other.

According to these results, hereafter, we only present the results of the best normalization method, i.e., FM visual-low-level recommendation technique based on log-quantile normalization, when performing comparison with the other recommender baselines.

4.2 Experiment 2: Feature Comparison

Table 2 presents the results we have obtained from the conducted experiments. As it can be seen, in terms of Precision, by far, the best technique is our proposed visual low-level feature based FM. Our technique obtained precision scores 0.0367, 0.0343, and 0.0286, while genre-based technique obtained scores of 0.0041, 0.0038 and 0.0040, for different recommendation size K at 5, 10, and 30. This result is promising since it shows that our technique based on automatic extraction of low-level visual features can achieve precision scores much better than genre-based recommendation.

Similar result has been observed w.r.t. recall metric. In terms of recall, our proposed technique, again similarly, have achieved the best result. While recall scores of our technique are 0.0272, 0.0488, 0.1176, genre-based FM obtained 0.0025, 0.0049, and 0.0170 for K at 5, 10, and 30. As expected, the non-personalized top-rated recommendation technique is the worst technique among all in terms of both precision and recall metrics.

We have also computed the F1 metric. Comparing the results, our proposed technique outperforms all the other technique in terms of F1. It achieves 0.0312, 0.0403, and 0.0461 scores and genre-based FM achieves 0.0031, 0.0043, and 0.0068 for K at 5, 10, and 30, which is substantially greater than the genre-based technique. Again top-rated technique achieves the worst result in terms of F1.

Comparing all these promising results, it is clear that our proposed technique, i.e., recommendation based on FM algorithm incorporating automatically extracted low-level visual features performs almost 10 times better scores than the recommendation based on rich source of expert-annotated genre labels, in terms of precision, recall , and F1 metrics.

5. CONCLUSION AND FUTURE WORK

This work presents a novel approach in the domain of content-based movie recommendations. The technique is based on the analysis of movie content and extraction of low-level visual features, fed to the Factorization Machine algorithm as side information, in order to generate personalized recommendations for users. This approach makes it possible to recommend items to users without relying on any high-level semantic features (e.g., genre) that are expensive to obtain, as they require expert level knowledge, and shall be missing (e.g., in new item scenario).

The results of our evaluation show that recommendations based on low-level visual features achieves almost 10 times better accuracy in comparison to the recommendations based on traditional set of high-level semantic features (i.e., genre).

For future work, we consider the design and development of an online web application in order to conduct online studies with real user. The goal is to evaluate the effectiveness of recommendations based on low-level visual features not only in terms of relevance, but also in terms of novelty, diversity and serendipity. Moreover, we will extend the range of low-level features extracted, and also, include audio features. Finally, we will extend the evaluation to user-generated videos.

6. ACKNOWLEDGMENTS

This work is supported by Telecom Italia S.p.A., Open Innovation Department, Joint Open Lab S-Cube, Milan.

7. REFERENCES

[1] Datasets | grouplens. http://grouplens.org/datasets/. Accessed: 2015-05-01.

[2] D. Bogdanov, J. Serrà, N. Wack, P. Herrera, and X. Serra. Unifying low-level and high-level music similarity measures. *Multimedia, IEEE Transactions on*, 13(4):687–701, 2011.

[3] B. M. Bolstad, R. A. Irizarry, M. Åstrand, and T. P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–193, 2003.

Table 1: Performance comparison of different normalizations of visual low-level features

Recommender	Norm	Precision			Recall			F1		
		@5	@10	@30	@5	@10	@30	@5	@10	@30
Visual-ll feature FM	**log-quantile**	**0.0367**	**0.0343**	**0.0286**	**0.0272**	**0.0488**	**0.1176**	**0.0312**	**0.0403**	**0.0461**
	quantile	0.0083	0.0081	0.0075	0.0052	0.0104	0.0306	0.0064	0.0091	0.0121
	log	0.0084	0.0084	0.0083	0.0051	0.0101	0.0308	0.0063	0.0092	0.0131

Table 2: Performance comparison of various recommendation techniques

Recommender	Precision			Recall			F1		
	@5	@10	@30	@5	@10	@30	@5	@10	@30
Visual-ll feature FM	**0.0367**	**0.0343**	**0.0286**	**0.0272**	**0.0488**	**0.1176**	**0.0312**	**0.0403**	**0.0461**
Genre-based FM	0.0041	0.0038	0.0040	0.0025	0.0049	0.0170	0.0031	0.0043	0.0068
Top rated	1.390e-05	1.042e-05	1.040e-05	1.922e-06	3.087e-06	1.111e-05	3.377e-06	4.764e-06	1.074e-05

[4] D. Bordwell, K. Thompson, and J. Ashton. *Film art: An introduction*, volume 7. McGraw-Hill New York, 1997.

[5] I. Cantador, M. Szomszor, H. Alani, M. Fernández, and P. Castells. Enriching ontological user profiles with tagging history for multi-domain recommendations. 2008.

[6] M. De Gemmis, P. Lops, C. Musto, F. Narducci, and G. Semeraro. Semantics-aware content-based recommender systems. In *Recommender Systems Handbook*, pages 119–159. Springer, 2015.

[7] M. De Gemmis, P. Lops, and G. Semeraro. A content-collaborative recommender that exploits wordnet-based user profiles for neighborhood formation. *User Modeling and User-Adapted Interaction*, 17(3):217–255, July 2007.

[8] Y. Deldjoo, M. Elahi, and P. Cremonesi. How to combine visual features with tags to improve the movie recommendation accuracy. In *International Conference on Electronic Commerce and Web Technologies*. Springer International Publishing, 2016.

[9] Y. Deldjoo, M. Elahi, P. Cremonesi, F. Garzotto, and P. Piazzolla. Recommending movies based on mise-en-scene design. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1540–1547. ACM, 2016.

[10] Y. Deldjoo, M. Elahi, P. Cremonesi, F. Garzotto, P. Piazzolla, and M. Quadrana. Content-based video recommendation system based on stylistic visual features. *Journal on Data Semantics*, pages 1–15, 2016.

[11] Y. Deldjoo, M. Elahi, M. Quadrana, and P. Cremonesi. Toward building a content-based video recommendation system based on low-level features. In *E-Commerce and Web Technologies*. Springer, 2015.

[12] M. Elahi, F. Ricci, and N. Rubens. A survey of active learning in collaborative filtering recommender systems. *Computer Science Review*, 2016.

[13] J. Gibbs. *Mise-en-scène: Film style and interpretation*, volume 10. Wallflower Press, 2002.

[14] L. Hong, A. S. Doumith, and B. D. Davison. Co-factorization machines: Modeling user interests and predicting individual decisions in twitter. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 557–566, New York, NY, USA, 2013. ACM.

[15] C. Musto, F. Narducci, P. Lops, G. Semeraro, M. de Gemmis, M. Barbieri, J. Korst, V. Pronk, and R. Clout. Enhanced semantic tv-show representation for personalized electronic program guides. In *User Modeling, Adaptation, and Personalization*, pages 188–199. Springer, 2012.

[16] M. Nasery, M. Elahi, and P. Cremonesi. Polimovie: a feature-based dataset for recommender systems. In *ACM RecSys Workshop on Crowdsourcing and Human Computation for Recommender Systems (CrawdRec)*, volume 3, pages 25–30. ACM, 2015.

[17] Z. Rasheed, Y. Sheikh, and M. Shah. On the use of computable features for film classification. *Circuits and Systems for Video Technology, IEEE Transactions on*, 15(1):52–64, 2005.

[18] S. Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, 2010.

[19] N. Rubens, M. Elahi, M. Sugiyama, and D. Kaplan. Active learning in recommender systems. In *Recommender Systems Handbook - chapter 24: Recommending Active Learning*, pages 809–846. Springer US, 2015.

[20] K. Seyerlehner, M. Schedl, T. Pohle, and P. Knees. Using block-level features for genre classification, tag classification and music similarity estimation. *Submission to Audio Music Similarity and Retrieval Task of MIREX 2010*, 2010.

[21] Y. Shi, M. Larson, and A. Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):3, 2014.

[22] B. Yang, T. Mei, X.-S. Hua, L. Yang, S.-Q. Yang, and M. Li. Online video recommendation based on multimodal fusion and relevance feedback. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 73–80. ACM, 2007.

[23] H. Zettl. Essentials of applied media aesthetics. In C. Dorai and S. Venkatesh, editors, *Media Computing*, volume 4 of *The Springer International Series in Video Computing*, pages 11–38. Springer US, 2002.

[24] X. Zhao, G. Li, M. Wang, J. Yuan, Z.-J. Zha, Z. Li, and T.-S. Chua. Integrating rich information for video recommendation with multi-task rank aggregation. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1521–1524. ACM, 2011.

Recommending Items with Conditions Enhancing User Experiences Based on Sentiment Analysis of Reviews

Konstantin Bauman
Stern School of Business
New York University
kbauman@stern.nyu.edu

Bing Liu
University of Illinois
at Chicago (UIC)
lub@cs.uic.edu

Alexander Tuzhilin
Stern School of Business
New York University
atuzhili@stern.nyu.edu

ABSTRACT

In this paper we propose a new method of recommending not only items of interest to the user but also the conditions enhancing user experiences with those items, such as recommending to go to a restaurant for seafood. This method is based on the sentiment analysis of user reviews, predicts sentiments that the user might express about the aspects determined in an application, and identifies the most valuable aspects of user's potential experience with the item. Furthermore, our method recommends the items together with those most important aspects over which the user has control and can potentially select them, such as the time to go to a restaurant, e.g. lunch vs. dinner, or what to have there, such as seafood. We tested our method on three applications (restaurants, hotels and beauty&spas) and experimentally showed that those users who followed our recommendations of items with their corresponding conditions had better experiences, as defined by the overall rating, than others.

CCS Concepts

•Information systems → Recommender systems; Sentiment analysis; •Computing methodologies → *Non-negative matrix factorization;*

Keywords

Recommender systems, user reviews, sentiment analysis, user experience, conditions of user experience

1. INTRODUCTION

Over the last five years, there has been much interest in leveraging user reviews for providing personalized recommendations based on these reviews [2]. Most of this work focuses on trying to improve estimations of user ratings based on the review and other relevant information [2] and also trying to explain why particular recommendations are provided to the user based on the review [15].

CBRecSys 2016, September 16, 2016, Boston, MA, USA.

These approaches aimed at predicting and explaining ratings in terms of the user and item characteristics without taking into the consideration additional factors, such as circumstances of consuming the item and personal choices of the user. For example, consider the user choosing between doing a manicure or a haircut in a salon. Depending on what the user would choose to do during her visit, she can give different ratings to the salon. Therefore, user experience of a particular item can be improved by recommending some appropriate conditions of consuming that item, such as doing a pedicure in that specific salon.

In this paper, we address this issue by recommending not only particular items, but also the *conditions* (i.e. the most important aspects) of potential experiences that the user can control, such as having haircut or pedicure in a salon. Furthermore, we can recommend certain actions to the management of an establishment (item) that would personalize experiences of the user when consuming the item (e.g., visiting the establishment). For example, we may recommend to the management of the Gotham Bar & Grill restaurant in New York to suggest the duck dish to a user because our method estimated that the user would particularly like that dish in that restaurant.

In this paper, we make the following contributions. First, we propose a novel approach of enhancing functionality of recommender systems by recommending not only the item itself but the item with the corresponding conditions enhancing user experience of the item. Further, we propose that the management of an establishment reviews the crucial aspects of the user's experience of the establishment that were identified by our method and uses them to personalize user experiences, thus improving the overall user ratings. Second, we developed a method for identifying the most valuable aspects of future experiences of the users that is based on the sentiment analysis of user reviews. Third, we tested our method on actual reviews and showed that users who followed our recommendations rate their experiences significantly higher than others.

2. LITERATURE REVIEW

Over the last few years, several papers tried to improve estimation of unknown ratings by using user reviews [5, 12, 16, 17]. For example, authors of [5] found six aspects in restaurant reviews and trained classifiers to identify them in text. Further they use this information to improve rating prediction quality. As another example, [11] uses the LDA-based approach combined with Matrix Factorization to better predict the unknown ratings. Furthermore, [3, 8]

use more extensive graphical models than [11] to predict unknown ratings that are based on collaborative filtering and topic modeling of user reviews. As a result they capture interpretable aspects and the sentiments of each aspect of a review. Moreover, [19] proposes another method to improve rating prediction based on learning users' interests and items' characteristics. In [18] authors proposed a tensor factorization approach for the rating inference. Operating on the tensor composed of the overall and the aspect ratings, this approach is able to capture the intrinsic relationships between users, items, and aspects of user reviews, and provide reasonably good predictions for unknown ratings. Finally, [10] determines aspect weight preferences of the user and estimates user satisfaction with the item based on these weights.

In contrast to these papers, in our work, we focus not only on rating predictions but also on *determining the most important aspects* having the highest impact on ratings measuring user's potential future experience of an item. Moreover, we provide recommendations not only of items of interest to the user but also the *conditions enhancing user experiences* with those items.

Furthermore, [1] constructed an aspect ontology for the Digital Camera application, developed a set of rules for identifying those aspects in text along with their sentiments. Based on the collected data, [1] aggregated profiles of these cameras and presented simple recommendations using knowledge based recommendation techniques. Also, [14] proposed an approach of extracting aspect-specific ratings from the reviews and then recommending new reviews to the users based on these ratings. In contrast to [1, 14], we focus on recommending the most important aspects over which the user has control and that she can potentially select.

Finally, there are a few publications on explanations of recommendations based on user reviews. In particular, [20] proposes the Explicit Factor Model (EFM) to generate explainable recommendations while keeping high prediction accuracy. In addition, [7] provides reasons why an item has been recommended vs. its various alternatives based on the aspects extracted from the reviews. In contrast to this work, our goal is not only to provide explanations but also to recommend the most important aspects over which the user has control and that she can potentially select.

3. OVERVIEW OF THE METHOD

In this section we present a method of identifying the most valuable aspects of future user experiences of the items and recommend the items together with the most important aspects over which the user has control and that she can potentially select. This method consists of sentiment analysis of user reviews, identification of relations between different aspects of the reviews, predicting sentiments that the user might express about these aspects, and calculating personal impact factors that each aspect contributes to the overall rating for the user. Moreover, we identify and recommend to the management of an establishment to consider potentially important aspects of the user experience with the establishment and use them to personalize user experiences there, thus improving the overall user rating of the establishment. The proposed method consists of 5 steps described below.

(1) Extracting aspects from the reviews.

In this step we utilized the state-of-the-art "industrial-strength" sentiment analysis system *Opinion Parser* [9] that uses Double Propagation method [13] for extracting aspects from texts. The sentiment classification algorithm is the lexicon-based method [9], which has its roots in [4, 6] but with significant improvements.

We applied *Opinion Parser* to the set of reviews R for a given application (e.g. restaurants) in order to build a set of aspects $\mathbb{A}$ occurring in R. Furthermore, for each review $r \in R$, *Opinion Parser* identifies a set of aspects A_r occurring in r and corresponding sentiments s^t_{ui} opinions of user u about aspects $t \in A_r$ of experience with item i.

(2) Training the sentiment predicting model.

In a typical review, set A_r contains only a small part of all the aspects $\mathbb{A}$, and thus for each particular aspect the matrix of known sentiments for (user, item) pairs is more sparse than matrix of ratings. Therefore, the problem of predicting sentiments of the aspects becomes even more challenging than prediction of the overall rating of review r.

Since various aspects of a review can be correlated, such as "desert" and "fruits", we propose an approach of using this correlation information to train the sentiment prediction model. In particular, in review r for each aspect $t \in \{\mathbb{A} - A_r\}$ we estimate its (unknown) sentiment s^t_{ui} as a weighted average of the explicitly specified sentiments of its k nearest neighbors. More formally,

$$\hat{s}^t_{ui} = \frac{\sum_1^k w_{tj} \cdot s^j_{ui}}{\sum_1^k w_{tj}},$$

where the weights w_{tj} are computed as Spearman correlations between aspects t and j. In case if the user did not express sentiments about a certain number of the correlated aspects, we leave the sentiment value of the aspect t blank.

Next, for each aspect $t \in \mathbb{A}$ we train the Matrix Factorization model in the following form:

$$\hat{s}^t_{ui} = \mu^t + b^t_u + b^t_i + p^t_u \cdot q^t_i.$$

This approach takes into account user's personal preferences, items individual characteristics and their interaction for each particular aspect $t \in \mathbb{A}$.

(3) Building regression model to predict ratings.

To build the regression model for predicting ratings, we first fill in the remaining missing values of sentiments for those aspects $t \in \{\mathbb{A} - A_r\}$ that were left blank in Step 2. We fill it with the average values of the aspects' sentiments $avg(s^t_{ui})$ taken across explicitly specified values in R. Note, that if the user did not mention the aspect in the review, we assume that she had the average satisfaction of the aspect and it did not affect her overall rating significantly.

We next build the regression model predicting rating r_{ui} of the review based on the estimated sentiments $\hat{s}^t_{ui}$ for aspects from $\mathbb{A}$. More specifically we estimate the overall rating with the regression model in the following form:

$$r_{ui} = (A + B_u + C_i) \cdot S_{ui} \quad (1)$$

where, $A = (a_0, \ldots, a_n)$ is a vector of general coefficients, $B_u = (b^u_0, \ldots, b^u_n)$ is a vector of coefficients pertaining to user u, $C_i = (c^i_0, \ldots, c^i_n)$ is a vector of coefficients pertaining to item i, and $S_{ui} = (\hat{s}^0_{ui}, \ldots, \hat{s}^n_{ui})$ is a vector of estimated values of sentiments corresponding to aspects from $\mathbb{A}$ in the particular review. Further, we avoid over-fitting by using the regularized model:

$$\min_{a_*, b_*, c_*} \sum_{(u,i) \in R} (r_{ui} - \overline{r}_{ui})^2 + \lambda \cdot \left(\sum_{t=0}^{n} \left(a_t^2 + \sum_u (b^u_t)^2 + \sum_i (c^i_t)^2 \right) \right)$$

As it follows from Eq.(1), the proposed model estimates individual preferences of user B_u and the individual characteristics of item C_i.

(4) Calculating impacts of aspects on rating.

In this step we apply the models built in Steps 2 and 3 for determining most important aspects of user's potential experiences with the establishment, where the importance of an aspect is determined by its weight in the regression model described in Step 3. More specifically, for a new potential review we predict sentiment $\overline{s}^t_{ui}$ of each of the aspects $t \in \mathbb{A}$. Then we compute the impact of each sentiment being potentially explicitly expressed in the review as follows. By construction, our regression model takes an average sentiment value in case when the user did not express her opinion. Therefore, we calculate the aspects' impact as product of the difference between the predicted value of sentiment and the average sentiment value for the particular aspect, with the corresponding coefficient in the regression model (Eq.(1)):

$$impact^t_{ui} = (a_t + b^u_t + c^i_t) \cdot (\overline{s}^t_{ui} - avg(s^t_{ui})). \qquad (2)$$

These calculated impacts reflect the importance of each aspect on the overall rating and they might be positive or negative.

(5) Recommending items and conditions.

Next, we identify two groups of aspects in $\mathbb{A}$ over which (a) the user has control and (b) the management of the establishment has control, in order to recommend most important of them in both cases. Furthermore, we identify the conditions that we want to recommend to the user or the management together with the item, where the condition is a suggestion to experience (positive) or not to experience (negative) a particular aspect. Finally, we recommend an item and its corresponding conditions to the user, or the most important conditions to the management.

For example, if our system identified aspect "fish" as having high *positive* impact on the rating, we will recommend this restaurant *and* the condition of ordering fish in that restaurant to the user. Similarly, if aspect "desert" has strong *negative* impact on the rating, we may still recommend to the user to visit that restaurant under the condition *not* to order desert there. Similarly, we can recommend such conditions to the management. For example, we can recommend to the management of a health&beauty salon to provide a complementary drink to the user (since it will improve her overall experience) and don't chat to her too much while in session.

In summary, we proposed a method for identifying and recommending the most valuable aspects of potential user experiences of the items. This method consists of sentiment analysis of user reviews, identification of relations between different aspects of the reviews, predicting sentiments that the user might express about these aspects, and calculating personal impact factors that each aspect contributes to the overall rating for the user. In Section 4, we show the results of applying the proposed method to real data from three applications.

4. EXPERIMENTS AND RESULTS

To demonstrate how well our method works in practice, we tested it on the Yelp dataset[1] for restaurants, hotels and beauty&spas applications for the reviews collected in several US cities over a period of 6 years. In this study we used the reviews of 24,917 restaurants produced by 384,821 users (1,344,405 reviews in total), 1,424 hotels produced by 65,387 users (96,384 reviews) and for the 6,536 beauty&spas produced by 71,422 users (104,199 reviews in total).

[1]https://www.yelp.com/dataset_challenge/dataset

Meat	Fish	Dessert	Money	Service	Decor
beef	cod	tiramisu	price	bartender	design
meat	salmon	cheesecake	dollars	waiter	ceiling
bbq	catfish	chocolate	cost	service	decor
ribs	tuna	dessert	budget	hostess	lounge
veal	shark	ice cream	charge	manager	window
pork	fish	macaroons	check	staff	space

Table 1: Examples of words pertaining to aspects in the restaurant application.

	Restaurants	*Hotels*	*Beauty & Spas*
Regression	1.256	1.275	1.343
Matrix Factorization	1.244	1.273	1.328

Table 2: RMSE of predicted ratings for our method vs. standard MF across three applications.

We applied the 5-step method presented in Section 3 to this Yelp data. As a result, we managed to extract 68 aspects for restaurants, 44 aspects for hotels, and 45 aspects for beauty&spas applications in Step 1 of our method using *Opinion Parser*. Table 1 presents several aspects pertaining to the restaurant application with examples of corresponding words.

Further, the set of reviews R in each application is partitioned into train and test sets in the ratio of 80% to 20%. After determining the sets of aspects in the reviews and aggregating their sentiments as described in Step 1, we filled in the missing sentiment values and trained the sentiment prediction models for each aspect, as described in Step 2.

We compared the performance of our approach with the standard MF models built only on the sets of explicitly expressed sentiments. Our results show that the proposed approach works better than standard one in terms of RMSE for most of the aspects across all the three applications. In particular, our method significantly outperformed standard MF for 43 aspects (out of 68) for restaurants, for 19 aspects (out of 44) for hotels, and for 33 aspects (out of 45) for beauty&spas. For *all* the remaining aspects, the differences between our method and the standard MF approach are not statistically significant. As it was expected, our approach works better for those aspects that have several close neighbors frequently mentioned in the reviews. For example, in a restaurant application aspect "music" is close to "atmosphere" and "interior", and therefore, our approach outperformed the standard one in predicting sentiments of the "music" aspect by 7.8% in terms of RMSE.

In Step 3 of our method, we obtained the rating prediction model and compared its performance with the standard MF approach to rating predictions. The results of this comparison (in terms of RMSE) are presented in Table 2, from which we conclude that the two methods are very similar in terms of their performance. This means that our regression model (Eq.(1)) predicts ratings reasonable well (comparably to the MF approach).

		Restaurants		*Hotels*		*Beauty & Spas*	
		users	*managers*	*users*	*managers*	*users*	*managers*
Positive Recommendations	Followed	3.818	3.816	3.410	3.537	4.176	4.167
	Other cases	3.734	3.737	3.320	3.324	4.051	4.053
Negative Recommendations	Not followed	3.482	3.473	3.105	2.869	3.740	3.744
	Other cases	3.784	3.787	3.342	3.429	4.126	4.127

Table 3: Average ratings for the users who followed (or not) our positive/negative recommendations of items with conditions.

We next applied our models to the test data and determined the most important aspects for each user and item pair, as described in Step 4. Finally, we produced recommendations both for the users and the management, as described in Step 5.

We next compared the ratings of the users and the management who followed our positive recommendations (i.e., mentioned the recommended aspects in their reviews) against other cases (i.e., that did not mention them) across the restaurants, hotels and beauty&spas applications. Further, we also did similar comparisons for negative recommendations (such as, do not order the desert in restaurant X), i.e., we compared users or managers who did not follow our negative recommendations (i.e., mentioned the negatively recommended aspects in their reviews) with others.

The results of these comparisons are presented in Table 3, where numbers in cells represent average ratings for users and managers across the three applications. The rows in Table 3 represent different conditions of whether or not the users followed our recommendations. As Table 3 shows, *our recommendations of items and conditions lead to higher evaluation ratings in those cases when users followed them* vs. other cases, and all the differences are statistically significant.

5. CONCLUSION

In this paper, we presented a new method of recommending not only items of interest to the user but also the conditions enhancing user experiences with those items. This method is based on the sentiment analysis of user reviews, predicts sentiments that the user might express about the aspects determined in an application, such as restaurants or hotels, and identifies the most valuable aspects of user's potential experience with the item. We tested it on three Yelp applications (restaurants, hotels and beauty&spas) and showed that our recommendations lead to higher evaluation ratings when users followed them vs. others.

This new approach to providing recommendations helps users to customize and enhance their experiences when consuming items, such as deciding what they should order in a particular restaurant.

6. REFERENCES

[1] S. Aciar, D. Zhang, S. Simoff, and J. Debenham. Informed recommender: Basing recommendations on consumer product reviews. *Intelligent Systems*, 2007.

[2] L. Chen, G. Chen, and F. Wang. Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction*, 2015.

[3] Q. Diao, M. Qiu, C.-Y. Wu, A. J. Smola, J. Jiang, and C. Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation. KDD, 2014.

[4] X. Ding, B. Liu, and P. S. Yu. A holistic lexicon-based approach to opinion mining. WSDM '08. ACM, 2008.

[5] G. Ganu, Y. Kakodkar, and A. Marian. Improving the quality of predictions using textual information in online user reviews. *Inf. Syst.*, 38(1):1–15, Mar. 2013.

[6] M. Hu and B. Liu. Mining and summarizing customer reviews. KDD '04. ACM, 2004.

[7] A. Lawlor, K. Muhammad, R. Rafter, and B. Smyth. Opinionated explanations for recommendation systems. *Research and Dev. in Int. System*, 2015.

[8] G. Ling, M. R. Lyu, and I. King. Ratings meet reviews, a combined approach to recommend. RecSys '14, pages 105–112, New York, NY, USA, 2014. ACM.

[9] B. Liu. *Sentiment Analysis and Opinion Mining.* Morgan and Claypool Publishers, 2012.

[10] H. Liu, J. He, T. Wang, W. Song, and X. Du. Combining user preferences and user opinions for accurate recommendation. *Electron. Commer. Rec. Appl.*, 2013.

[11] J. McAuley and J. Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. RecSys '13. ACM, 2013.

[12] J. McAuley, J. Leskovec, and D. Jurafsky. Learning attitudes and attributes from multi-aspect reviews. ICDM '12, pages 1020–1025, 2012.

[13] G. Qiu, B. Liu, J. Bu, and C. Chen. Opinion word expansion and target extraction through double propagation. *Comput. Linguist.*, 2011.

[14] V. Suresh, S. Roohi, and M. Eirinaki. Aspect-based opinion mining and recommendationsystem for restaurant reviews. RecSys '14. ACM, 2014.

[15] N. Tintarev and J. Masthoff. Explaining recommendations: Design and evaluation. *Recommender Systems Handbook 2nd edt.*, 2015.

[16] I. Titov and R. McDonald. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL-08: HLT*, 2008.

[17] H. Wang, Y. Lu, and C. Zhai. Latent aspect rating analysis without aspect keyword supervision. KDD '11. ACM, 2011.

[18] Y. Wang, Y. Liu, and X. Yu. Collaborative filtering with aspect-based opinion mining: A tensor factorization approach. In *IEEE ICDM*, 2012.

[19] Y. Xu, Z. Chen, J. Yin, Z. Wu, and T. Yao. Learning to recommend with user generated content. *WAIM*, 2015.

[20] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *ACM SIGIR*, 2014.

RDF Graph Embeddings for Content-based Recommender Systems

Jessica Rosati[1,2]
[1]University of Camerino – Piazza Cavour 19/f – 62032 Camerino, Italy
[2]Polytechnic University of Bari – Via Orabona, 4 – 70125 Bari, Italy
jessica.rosati@unicam.it

Petar Ristoski
Data and Web Science Group, University of Mannheim, B6, 26, 68159 Mannheim, Germany
petar.ristoski@informatik.uni-mannheim.de

Tommaso Di Noia
Polytechnic University of Bari – Via Orabona, 4 – 70125 Bari, Italy
tommaso.dinoia@poliba.it

Renato De Leone
University of Camerino – Piazza Cavour 19/f – 62032 Camerino, Italy
renato.deleone@unicam.it

Heiko Paulheim
Data and Web Science Group, University of Mannheim, B6, 26, 68159 Mannheim, Germany
heiko@informatik.uni-mannheim.de

ABSTRACT

Linked Open Data has been recognized as a useful source of background knowledge for building content-based recommender systems. Vast amount of RDF data, covering multiple domains, has been published in freely accessible datasets. In this paper, we present an approach that uses language modeling approaches for unsupervised feature extraction from sequences of words, and adapts them to RDF graphs used for building content-based recommender system. We generate sequences by leveraging local information from graph sub-structures and learn latent numerical representations of entities in RDF graphs. Our evaluation on two datasets in the domain of movies and books shows that feature vector representations of general knowledge graphs such as DBpedia and Wikidata can be effectively used in content-based recommender systems.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval

Keywords

Recommender System; Graph Embeddings; Linked Open Data

1. INTRODUCTION

One of the main limitations of traditional content-based recommendation approaches is that the information on which they rely is generally insufficient to elicit user's interests and characterize all the aspects of her interaction with the system. This is the main drawback of the approaches built on textual and keyword-based representations, which cannot capture complex relations among objects since they lack the semantics associated to their attributes. A process of "knowledge infusion" [40] and semantic analysis has been proposed to face this issue, and numerous approaches that incorporate ontological knowledge have been proposed, giving rise to the newly defined class of *semantics-aware* content-based recommender systems [6]. More recently the *Linked Open Data* (LOD) initiative [3] has opened new interesting possibilities to realize better recommendation approaches. The LOD initiative in fact gave rise to a variety of open knowledge bases freely accessible on the Web and being part of a huge decentralized knowledge base, the LOD cloud, where each piece of little knowledge is enriched by links to related data. LOD is an open, interlinked collection of datasets in machine-interpretable form, built on *World Wide Web Consortium* (W3C) standards as RDF[1], and SPARQL[2]. Currently the LOD cloud consists of about $1,000$ interlinked datasets covering multiple domains from life science to government data [39]. It has been shown that LOD is a valuable source of background knowledge for content-based recommender systems in many domains [12]. Given that the items to be recommended are linked to a LOD dataset, information from LOD can be exploited to determine which items are considered to be similar to the ones that the user has consumed in the past, allowing to discover hidden information and implicit relations between objects [26]. While LOD is rich in high quality data, it is still challenging to find effective and efficient way of exploiting the knowledge for content-based recommendations. So far, most of the pro-

CBRecSys 2016, September 16, 2016, Boston, MA, USA.

[1]http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, 2004.
[2]http://www.w3.org/TR/rdf-sparql-query/, 2008

posed approaches in the literature are supervised or semi-supervised, which means cannot work without human interaction.

In this work, we adapt language modeling approaches for latent representation of entities in RDF graphs. To do so, we first convert the graph into a set of sequences of entities using graph walks. In the second step, we use those sequences to train a neural language model, which estimates the likelihood of a sequence of entities appearing in the graph. Once the training is finished, each entity in the graph is represented with a vector of latent numerical values. Projecting such latent representation of entities into a lower dimensional feature space shows that semantically similar entities appear closer to each other. Such entity vectors can be directly used in a content-based recommender system.

In this work, we utilize two of the most prominent RDF knowledge graphs [29], i.e. DBpedia [18] and Wikidata [42]. DBpedia is a knowledge graph which is extracted from structured data in Wikipedia. The main source for this extraction are the key-value pairs in the Wikipedia infoboxes. Wikidata is a collaboratively edited knowledge graph, operated by the Wikimedia foundation[3] that also hosts various language editions of Wikipedia.

The rest of this paper is structured as follows. In Section 2, we give an overview of related work. In Section 3, we introduce our approach, followed by an evaluation in Section 4. We conclude with a summary and an outlook on future work.

2. RELATED WORK

It has been shown that LOD can improve recommender systems towards a better understanding and representation of user preferences, item features, and contextual signs they deal with. LOD has been used in content-based, collaborative, and hybrid techniques, in various recommendation tasks, i.e., rating prediction, top-N recommendations and improving of diversity in content-based recommendations. LOD datasets, e.g. DBpedia, have been used in content-based recommender systems in [11] and [12]. The former performs a semantic expansion of the item content based on ontological information extracted from DBpedia and LinkedMDB [16], the first open semantic web database for movies, and tries to derive implicit relations between items. The latter involves DBpedia and LinkedMDB too, but is an adaptation of the Vector Space Model to Linked Open Data: it represents the RDF graph as a 3-dimensional tensor where each slice is an ontological property (e.g. starring, director,...) and represents its adjacency matrix. It has been proven that leveraging LOD datasets is also effective for hybrid recommender systems [4], that is in those approaches that boost the collaborative information with additional knowledge, such as the item content. In [10] the authors propose *SPRank*, a hybrid recommendation algorithm that extracts semantic path-based features from DBpedia and uses them to compute top-N recommendations in a learning to rank approach and in multiple domains, movies, books and musical artists. SPRank is compared with numerous collaborative approaches based on matrix factorization [17, 34] and with other hybrid RS, such as *BPR-SSLIM* [25], and exhibits good performance especially in those contexts characterized by high sparsity, where the contribution of the content becomes essential. Another hybrid approach is proposed in [36], which builds on training individual base recommenders and using global popularity scores as generic recommenders. The results of the individual recommenders are combined using stacking regression and rank aggregation. Most of these approaches can be referred to as *top-down* approaches [6], since they rely on the integration of external knowledge and cannot work without human intervention. On the other side, *bottom-up* approaches ground on the *distributional hypothesis* [15] for language modeling, according to which the meaning of words depends on the context in which they occur, in some textual content. The resulting strategy is therefore unsupervised, requiring a corpora of textual documents for training as large as possible. Approaches based on the distributional hypothesis, referred to as *discriminative models*, behave as word embeddings techniques where each term (and document) becomes a point in the vector space. They substitute the term-document matrix typical of Vector Space Model with a term-context matrix on which they apply dimensionality reduction techniques such as Latent Semantic Indexing (LSI) [8] and the more scalable and incremental Random Indexing (RI) [38]. The latter has been involved in [22] and [23] to define the so called enhanced Vector Space Model (eVSM) for content-based RS, where user's profile is incrementally built summing the features vectors representing documents liked by the user and a negation operator is introduced to take into account also negative preferences.

Word embedding techniques are not limited to LSI and RI. The word2vec strategy has been recently presented in [19] and [20], and to the best of our knowldge, has been applied to item recommendations in a few works [21, 28]. In particular, [21] is an empirical evaluation of LSI, RI and word2vec to make content-based movie recommendation exploiting textual information from Wikipedia, while [28] deals with check-in venue (location) recommendations and adds a non-textual feature, the past check-ins of the user. They both draw the conclusion that word2vec techniques are promising for the recommendation task. Finally there is a single example of *product embedding* [14], namely *prod2vec*, which operates on the artificial graph of purchases, treating a purchase sequence as a "sentence" and products within the sequence as words.

3. APPROACH

In our approach, we adapt neural language models for RDF graph embeddings. Such approaches take advantage of the word order in text documents, explicitly modeling the assumption that closer words in the word sequence are statistically more dependent. In the case of RDF graphs, we follow the approach sketched in [37], considering entities and relations between entities instead of word sequences. Thus, in order to apply such approaches on RDF graph data, we have to transform the graph data into sequences of entities, which can be considered as sentences. After the graph is converted into a set of sequences of entities, we can train the same neural language models to represent each entity in the RDF graph as a vector of numerical values in a latent feature space. Such entity vectors can be directly used in a content-based recommender system.

3.1 RDF Graph Sub-Structures Extraction

We propose random graph walks as an approach for con-

[3]http://wikimediafoundation.org/

verting graphs into a set of sequences of entities.

DEFINITION 1. *An RDF graph is a graph G = (V, E), where V is a set of vertices, and E is a set of directed edges.*

The objective of the conversion functions is for each vertex $v \in V$ to generate a set of sequences S_v, where the first token of each sequence $s \in S_v$ is the vertex v followed by a sequence of tokens, which might be edges, vertices, or any substructure extracted from the RDF graph, in an order that reflects the relations between the vertex v and the rest of the tokens, as well as among those tokens.

In this approach, for a given graph $G = (V, E)$, for each vertex $v \in V$ we generate all graph walks P_v of depth d rooted in the vertex v. To generate the walks, we use the breadth-first algorithm. In the first iteration, the algorithm generates paths by exploring the direct outgoing edges of the root node v_r. The paths generated after the first iteration will have the following pattern v_r ->e_{1i}, where $i \in E(v_r)$. In the second iteration, for each of the previously explored edges the algorithm visits the connected vertices. The paths generated after the second iteration will follow the following pattern v_r ->e_{1i} ->v_{1i}. The algorithm continues until d iterations are reached. The final set of sequences for the given graph G is the union of the sequences of all the vertices $\bigcup_{v \in V} P_v$.

3.2 Neural Language Models – word2vec

Until recently, most of the Natural Language Processing systems and techniques treated words as atomic units, representing each word as a feature vector using a one-hot representation, where a word vector has the same length as the size of a vocabulary. In such approaches, there is no notion of semantic similarity between words. While such approaches are widely used in many tasks due to their simplicity and robustness, they suffer from several drawbacks, e.g., high dimensionality and severe data sparsity, which limit the performance of such techniques. To overcome such limitations, neural language models have been proposed, inducing low-dimensional, distributed embeddings of words by means of neural networks. The goal of such approaches is to estimate the likelihood of a specific sequence of words appearing in a corpus, explicitly modeling the assumption that closer words in the word sequence are statistically more dependent.

While some of the initially proposed approaches suffered from inefficient training of the neural network models, with the recent advancements in the field several efficient approaches has been proposed. One of the most popular and widely used is the word2vec neural language model [19, 20]. Word2vec is a particularly computationally-efficient two-layer neural net model for learning word embeddings from raw text. There are two different algorithms, the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model.

3.2.1 Continuous Bag-of-Words Model

The CBOW model predicts target words from context words within a given window.The input layer is comprised from all the surrounding words for which the input vectors are retrieved from the input weight matrix, averaged, and projected in the projection layer. Then, using the weights from the output weight matrix, a score for each word in the vocabulary is computed, which is the probability of the word being a target word. Formally, given a sequence of training words $w_1, w_2, w_3, ..., w_T$, and a context window c, the objective of the CBOW model is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T} log\ p(w_t|w_{t-c}...w_{t+c}), \quad (1)$$

where the probability $p(w_t|w_{t-c}...w_{t+c})$ is calculated using the softmax function:

$$p(w_t|w_{t-c}...w_{t+c}) = \frac{exp(\bar{v}^T v'_{w_t})}{\sum_{w=1}^{V} exp(\bar{v}^T v'_w)}, \quad (2)$$

where v'_w is the output vector of the word w, V is the complete vocabulary of words, and $\bar{v}$ is the averaged input vector of all the context words:

$$\bar{v} = \frac{1}{2c}\sum_{-c \le j \le c, j \ne 0} v_{w_{t+j}} \quad (3)$$

3.2.2 Skip-Gram Model

The Skip-Gram model does the inverse of the CBOW model and tries to predict the context words from the target words. More formally, given a sequence of training words $w_1, w_2, w_3, ..., w_T$, and a context window c, the objective of the skip-gram model is to maximize the following average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c \le j \le c, j \ne 0} log\ p(w_{t+j}|w_t), \quad (4)$$

where the probability $p(w_{t+j}|w_t)$ is calculated using the softmax function:

$$p(w_o|w_i) = \frac{exp(v'^T_{wo} v_{wi})}{\sum_{w=1}^{V} exp(v'^T_w v_{wi})}, \quad (5)$$

where v_w and v'_w are the input and the output vector of the word w, and V is the complete vocabulary of words.

In both cases, calculating the softmax function is computationally inefficient, as the cost for computing is proportional to the size of the vocabulary. Therefore, two optimization techniques have been proposed, i.e., hierarchical softmax and negative sampling [20]. The empirical studies show that in most cases negative sampling leads to better performances than hierarchical softmax, which depends on the selected negative samples, but it has higher runtime.

Once the training is finished, semantically similar words appear close to each other in the feature space. Furthermore, basic mathematical functions can be performed on the vectors, to extract different relations between the words.

4. EVALUATION

We evaluate different variants of our approach on two distinct datasets, and compare them to common approaches for creating content-based item representations from LOD and with state of the art collaborative approaches. Furthermore, we investigate the use of two different LOD datasets as background knowledge, i.e., DBpedia and Wikidata.

4.1 Datasets

In order to test the effectiveness of our proposal, we evaluate it in terms of ranking accuracy and aggregate diversity on two datasets belonging to different domains, i.e. `Movielens` 1M[4] for movies and `LibraryThing`[5] for books. The

[4]http://grouplens.org/datasets/movielens/
[5]https://www.librarything.com/

former contains 1 million 1-5 stars ratings from 6,040 users on 3,883 movies. The `LibraryThing` dataset contains more than 2 millions ratings from 7,564 users on 39,515 books. As there are many duplicated ratings in the dataset, when a user has rated more than once the same item, we select her last rating. This choice brings to have 626,000 ratings in the range from 1 to 10. The user-item interactions contained in the datasets are enriched with side information thanks to the item mapping and linking to DBpedia technique detailed in [27], whose dump is available at http://sisinflab.poliba.it/semanticweb/lod/recsys/datasets/. In the attempt to reduce the popularity bias from our final evaluation we decided to remove the top 1% most popular items from both datasets [5]. Moreover we keep out, from `LibraryThing`, users with less than five ratings and items rated less than five times, and to have a dataset characterized by lower sparsity we retain for `Movielens` only users with at least fifty ratings, as already done in [10]. Table 1 contains the final statistics for our datasets.

	Movielens	**LibraryThing**
Number of users	4,186	7,149
Number of items	3,196	4,541
Number of ratings	822,597	352,123
Data sparsity	93.85%	98.90%

Table 1: Statistics about the two datasets

4.1.1 RDF Embeddings

As RDF datasets we use DBpedia and Wikidata.

We use the English version of the 2015-10 DBpedia dataset, which contains $4,641,890$ instances and $1,369$ mapping-based properties. In our evaluation we only consider object properties, and ignore the data properties and literals.

For the Wikidata dataset we use the simplified and derived RDF dumps from 2016-03-28[6]. The dataset contains $17,340,659$ entities in total. As for the DBpedia dataset, we only consider object properties, and ignore the data properties and literals.

4.2 Evaluation Protocol

As evaluation protocol for our comparison, we adopted the *all unrated items* methodology presented in [41] and already used in [10]. Such methodology asks to predict a score for each item not rated by a user, irrespective of the existence of an actual rating, and to compare the recommendation list with the test set.

The metrics involved in the experimental comparison are precision, recall and nDCG as accuracy metrics, and catalog coverage and Gini coefficient for the aggregate diversity. *precision@N* represents the fraction of relevant items in the top-N recommendations. *recall@N* indicates the fraction of relevant items, in the user test set, occurring in the top-N list. As relevance threshold, we set 4 for `Movielens` and 8 for `LibraryThing`, as previously done in [10]. Although precision and recall are good indicators to evaluate the accuracy of a recommendation engine, they are not rank-sensitive. *nDCG@N* [2] instead takes into account also the position in the recommendation list, being defined as

$$\text{nDCG@}N = \frac{1}{\text{iDCG}} \cdot \sum_{i=1}^{N} \frac{2^{rel(u,i)} - 1}{log_2(1+i)} \tag{6}$$

where $rel(u,i)$ is a boolean function representing the relevance of item i for user u and iDCG is a normalization factor that sets nDCG@N value to 1 when an ideal ranking is returned [2]. As suggested in [41] and set up in [10], in the computation of nDCG@N we fixed a default "neutral" value for those items with no ratings, i.e. 3 for `Movielens` and 5 for `LibraryThing`.

Providing accurate recommendations has been recognized as just one of the main task a recommender system must be able to perform. We therefore evaluate the contribution of our latent features in terms of aggregate diversity, and more specifically by means of catalog coverage and Gini coefficient [1]. The *catalog coverage* represents the percentage of available candidate items recommended at least once. It is an important quality dimension for both user and business perspective [13], since it exhibits the capacity to not settle just on a subset of items (e.g. the most popular). This metric however should be supported by a distribution metric which has to show the ability of a recommendation engine to equally spread out the recommendations across all users. *Gini coefficient* [1] is used for this purpose, since it measures the concentration degree of top-N recommendations across items and is defined as

$$\text{Gini} \quad = \quad 2\sum_{i=1}^{n} \left(\frac{n+1-i}{n+1}\right) \cdot \left(\frac{rec(i)}{total}\right) \tag{7}$$

In Equation (7), n is the number of candidate items available for recommendation, *total* represents the total number of top-N recommendations made across all users, and $rec(i)$ is the number of users to whom item i has been recommended. Gini coefficient gives therefore an idea of the "equity" in the distribution of the items. It is worth to remind that we are following the notion given in [1], where the complement of the standard Gini coefficient is used, so that higher values correspond to more balanced recommendations.

4.3 Experimental Setup

The first step of our approach is to convert the RDF graphs into a set of sequences. Therefore, to extract the entities embeddings for the large RDF datasets, we use only random graph walks entity sequences. More precisely, we follow the approach presented in [32] to generate only a limited number of random walks for each entity. For DBpedia, we experiment with 500 walks per entity with depth of 4 and 8, while for Wikidata, we use only 200 walks per entity with depth of 4. Additionally, for each entity in DBpedia and Wikidata, we include all the walks of depth 2, i.e., direct outgoing relations. We use the corpora of sequences to build both CBOW and Skip-Gram models with the following parameters: window size = 5; number of iterations = 5; negative sampling for optimization; negative samples = 25; with average input vector for CBOW. We experiment with 200 and 500 dimensions for the entities' vectors. All the models are publicly available[7].

We compare our approach to several baselines. For generating the data mining features, we use three strategies that

[6]http://tools.wmflabs.org/wikidata-exports/rdf/index.php?content=dump_download.php\&dump=20160328

[7]http://data.dws.informatik.uni-mannheim.de/rdf2vec/

take into account the direct relations to other resources in the graph [30], and two strategies for features derived from graph sub-structures [7]:

- Features derived from specific relations. In the experiments we use the relations *rdf:type* (types), and *dcterms:subject* (categories) for datasets linked to DBpedia.
- Features derived from generic relations, i.e., we generate a feature for each incoming (rel in) or outgoing relation (rel out) of an entity, ignoring the value of the relation.
- Features derived from generic relations-values, i.e, we generate feature for each incoming (rel-vals in) or outgoing relation (rel-vals out) of an entity including the value of the relation.
- Kernels that count substructures in the RDF graph around the instance node. These substructures are explicitly generated and represented as sparse feature vectors.
  - The Weisfeiler-Lehman (WL) graph kernel for RDF [7] counts full subtrees in the subgraph around the instance node. This kernel has two parameters, the subgraph depth d and the number of iterations h (which determines the depth of the subtrees). We use $d = 1$ and $h = 2$ and therefore we will indicate this strategy as WL12.
  - The Intersection Tree Path kernel for RDF [7] counts the walks in the subtree that span from the instance node. Only the walks that go through the instance node are considered. We will therefore refer to it as the root Walk Count (WC) kernel. The root WC kernel has one parameter: the length of the paths l, for which we test 2. This strategy will be denoted accordingly as WC2.

The strategies for creating propositional features from Linked Open Data are implemented in the RapidMiner LOD extension[8] [31, 35].

4.4 Results

The target of the experimental section of this paper is two-fold. On the one hand, we want to prove that the latent features we extracted are able to subsume the other kind of features in terms of accuracy and aggregate diversity. On the other hand we aim at qualifying our strategies as valuable means for the recommendation task, through a first comparison with state of the art approaches. Both goals are pursued implementing an item-based K-nearest-neighbor method, hereafter denoted as ItemKNN, with cosine similarity among features vectors. Formally, this method determines similarities between items through cosine similarity between relative vectors and then selects a subset of them – the neighbors – for each item, that will be used to estimate the rating of user u for a new item i as follows:

$$r^*(u,i) = \sum_{j \in ratedItems(u)} cosineSim(j,i) \cdot r_{u,j}$$

where $ratedItems(u)$ is the set of items already evaluated by user u, $r_{u,j}$ indicates the rating for item j by user u and $cosineSim(j,i)$ is the cosine similarity score between items j and i. In our experiments, the size of the considered neighbourhood is limited to 5. The computation of recommendations has been done with the publicly available library RankSys[9]. All the results have been computed @10, that is considering the top-10 lists recommended to the users: precision, recall and nDCG are computed for each user and then averaged across all users, while diversity metrics are global measures.

Tables 2 and 3 contain the values of precision, recall and nDCG, respectively for `Movielens` and `LibraryThing`, for each kind of features we want to test. The best approach for both datasets is retrieved with a Skip-Gram model and with a size of 200 for vectors built upon DBpedia. For the sake of truth, on the `Movielens` dataset the highest value of precision is achieved using vector size of 500, but the size 200 is prevalent according to the F1 measure, i.e. the harmonic mean of precision and recall. A substantial difference however concerns the exploratory depth of the random walks, since for `Movielens` the results related to depth 4 outdo those computed with depth 8, while the tendency is reversed for `LibraryThing`. The advantage of the Skip-Gram model over the CBOW is a constant both on DBpedia and Wikidata. Moreover, the employment of the Wikidata RDF dataset turns out to be more effective for `LibraryThing`, where the Skip-Gram vectors with depth 4 exceeds the corresponding DBpedia vectors. Moving to the features extracted from direct relations, the contribution of the "categories" stands clearly out, together with relations-values "rel-vals", especially when just incoming relations are considered. The extraction of features from graph structures, i.e. WC2 and WL12 approaches, seems not to provide significant advantages to the recommendation algorithm.

To point out that our latent features are able to capture the structure of the RDF graph, placing closely semantically similar items, we provide some examples of the neighbouring sets retrieved using our graph embeddings technique and used within the ItemKNN. Table 4 is related to movies and displays that neighboring items are highly relevant and close to the query item, i.e. the item for which neighbors are searched for.

To further analyse the semantics of the vector representations, we employ Principal Component Analysis (PCA) to project the "high"-dimensional entities' vectors in a two dimensional feature space, or 2D scatter plot. For each of the query movies in Table 4 we visualize the vectors of the 5 nearest neighbors as shown in Figure 1. The figure illustrates the ability of the model to automatically cluster the movies.

The impact on the aggregate diversity. As a further validation of the interactiveness of our latent features for recommendation task, we report the performances of the ItemKNN approach in terms of aggregate diversity. The relation between accuracy and aggregate diversity has gained the attention of researchers in the last few years and is generally characterized as a trade-off [1]. Quite surprisingly, however, the increase in accuracy, shown in Tables 2 and 3, seems not to rely on a concentration on a subset of items, e.g. the most

[8]http://dws.informatik.uni-mannheim.de/en/research/rapidminer-lod-extension

[9]http://ranksys.org/

Strategy	P@10	R@10	nDCG@10
DB2vec CBOW 200 4	0.03893	0.02167	0.30782
DB2vec CBOW 500 4	0.03663	0.02088	0.30557
DB2vec SG 200 4	0.05681	**0.03119**	**0.31828**
DB2vec SG 500 4	**0.05786**	0.0304	0.31726
DB2vec CBOW 200 8	0.01064	0.00548	0.29245
DB2vec CBOW 500 8	0.01137	0.00567	0.29289
DB2vec SG 200 8	0.04424	0.02693	0.30997
DB2vec SG 500 8	0.02191	0.01478	0.29863
WD2vec CBOW 200 4	0.01217	0.00596	0.29362
WD2vec CBOW 500 4	0.01027	0.00427	0.29211
WD2vec SG 200 4	0.02902	0.01479	0.30189
WD2vec SG 500 4	0.02644	0.01246	0.29967
types	0.00313	0.00145	0.28864
categories	0.0305	0.02093	0.30444
rel in	0.01122	0.00589	0.29183
rel out	0.02844	0.01607	0.30274
rel in & out	0.02852	0.01566	0.3006
rel-vals in	0.03883	0.02293	0.29411
rel-vals out	0.01279	0.00971	0.29378
rel-vals in & out	0.01174	0.00913	0.29333
WC2	0.00684	0.00343	0.29032
WL12	0.00601	0.00288	0.28977

Table 2: Results of the ItemKNN approach on `Movielens` dataset. P and R stand respectively for precision and recall, SG indicates the Skip-Gram model, and DB and WD represent DBpedia and Wikidata respectively.

Strategy	P@10	R@10	nDCG@10
DB2vec CBOW 200 4	0.05127	0.11777	0.21244
DB2vec CBOW 500 4	0.05065	0.11557	0.21039
DB2vec SG 200 4	0.05719	0.12763	0.2205
DB2vec SG 500 4	0.05811	0.12864	0.22116
DB2vec CBOW 200 8	0.00836	0.02334	0.14147
DB2vec CBOW 500 8	0.00813	0.02335	0.14257
DB2vec SG 200 8	**0.07681**	**0.17769**	**0.25234**
DB2vec SG 500 8	0.07446	0.1743	0.24809
WD2vec CBOW 200 4	0.00537	0.01084	0.13524
WD2vec CBOW 500 4	0.00444	0.00984	0.13428
WD2vec SG 200 4	0.06416	0.14565	0.23309
WD2vec SG 500 4	0.06031	0.14194	0.22752
types	0.01854	0.04535	0.16064
categories	0.06662	0.15258	0.23733
rel in	0.04577	0.10219	0.20196
rel out	0.04118	0.09055	0.19449
rel in & out	0.04531	0.10165	0.20115
rel-vals in	0.06176	0.14101	0.22574
rel-vals out	0.06163	0.13763	0.22826
rel-vals in & out	0.06087	0.13662	0.22615
WC2	0.00159	0.00306	0.12858
WL12	0.00155	0.00389	0.12937

Table 3: Results of the ItemKNN approach on `LibraryThing` dataset.

popular ones, according to the results proposed in Tables 5 and 6. Here we are reporting, for the sake of concisenesses, only the best approaches for each kind of features. More clearly, we are displaying the best approach for latent features computed on DBpedia, the best approach for latent features computed on Wikidata and the values for the strategy involving categories, since it provides the highest scores among features extracted through direct relations. We are not reporting the values related to WL12 and WC2 algorithms, since their contribution is rather low also in this

Query Movie	K Nearest Neighbours
Batman	Batman Forever, Batman Returns, Batman & Robin, Superman IV: The Quest for Peace, Dick Tracy
Bambi	Cinderella, Dumbo, 101 Dalmatians , Pinocchio, Lady and the Tramp
Star Trek: Generations	Star Trek VI: The Undiscovered Country, Star Trek: Insurrection, Star Trek III: The Search for Spock, Star Trek V: The Final Frontier, Star Trek: First Contact (1996)

Table 4: Examples of K-nearest-neighbor sets on `Movielens`, for the Skip-Gram model with depth of 4 and size vectors 200, on DBpedia.

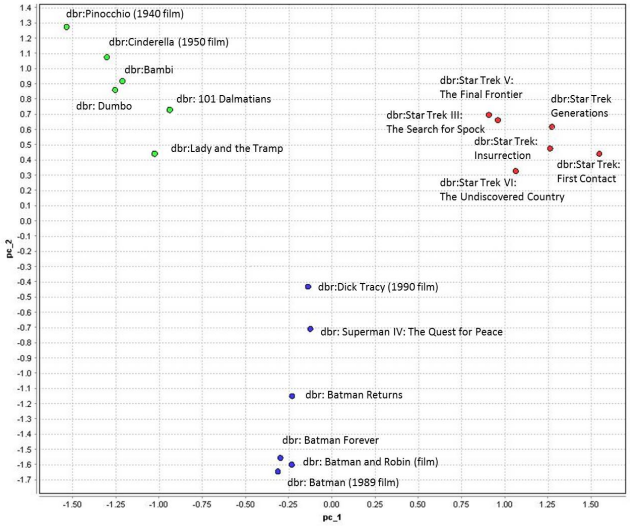


Figure 1: Two-dimensional PCA projection of the 200-dimensional Skip-gram vectors of movies in Table 4.

analysis. For both movies and books domain, the best approaches found on DBpedia for the accuracy metrics, i.e. respectively "DB2vec SG 200 4" and "DB2vec SG 200 8", perform better also in terms of aggregate diversity. For the `LibraryThing` dataset the Skip-Gram model computed with random walks on Wikidata and size vector limited to 200 is very close to the highest scores retrieved in DBpedia, while for `Movielens` is the CBOW model, with depth 4, to gain the best performance on Wikidata. The contribution of the categories, despite being lower than the best approach on each dataset, is quite significant for diversity measures too.

Comparison with state of the art collaborative approaches. It is a quite common belief in the RS field that using pure content-based approaches would not be enough to provide accurate suggestions and that the recommendation engines must ground on collaborative information too. This motivated us to explicitly compare the best approaches built on graph embeddings technique with the well-known state of the art collaborative recommendation algorithms listed be-

Strategy	Coverage	Gini
DB2vec SG 200 4	**0.35198**	**0.07133**
WD2vec CBOW 200 4	0.27749	0.04052
categories	0.29798	0.04714

Table 5: Methods comparison in terms of aggregate diversity on the `Movielens` dataset. Coverage stands for catalog coverage and Gini for Gini coefficient.

Strategy	Coverage	Gini
DB2vec SG 200 8	**0.76386**	**0.29534**
WD2vec SG 200 4	0.73037	0.28525
categories	0.7246	0.26409

Table 6: Methods comparison in terms of aggregate diversity on the `LibraryThing` dataset.

low, and implemented with the publicly available software library MyMediaLite[10].

- Biased Matrix Factorization (MF) [17], recognized as the state of the art for rating prediction, is a matrix factorization model that minimizes RMSE using stochastic gradient descent and both user and item bias.
- PopRank is a baseline based on popularity. It recommends the same recommendations to all users according to the overall items popularity. Recent studies have point out that recommending the most popular items could already result in a high performance [5].
- Bayesian Personalized Ranking (BPRMF) combines a matrix factorization approach with a Bayesian Personalized Ranking optimization criterion [34].
- SLIM [24] is a Sparse LInear Method for top-N recommendation that learns a sparse coefficient matrix for the items involved in the system by only relying on the users purchase/ratings profile and by solving a L1-norm and L2-norm regularized optimization problem.
- Soft Margin Ranking Matrix Factorization (RankMF) is a matrix factorization approach for ranking, whose loss function is ordinal regression [43].

Tables 7 and 8 provide the comparison results for `Movielens` and `LibraryThing` respectively. Table 7 shows that matrix factorization techniques and the SLIM algorithm exceed our approach based only on content information. This outcome was somehow expected, especially considering that, in our experimental setting, `Movielens` dataset retains only users with at least fifty ratings. The community-based information is unquestionably predominant for this dataset, whose sparsity would probably be unlikely for most real-world scenarios. The behaviour however is completely overturned on the `LibraryThing` dataset, whose results are collected in Table 8. In this case, the mere use of our features vectors (i.e. the "DB2vec SG 200 8" strategy) is able to outperform the competitor algorithms, which are generally regarded as the most efficient collaborative algorithms for both rating and ranking prediction.

[10]http://www.mymedialite.net

Strategy	P@10	R@10	nDCG@10
DB2vec SG 200 4	0.0568	0.0312	0.3183
MF	0.2522	0.1307	0.4427
PopRank	0.1673	0.0787	0.3910
BPRMF	0.2522	0.1307	0.4427
SLIM	**0.2632**	**0.1474**	**0.4599**
RankMF	0.1417	0.0704	0.3736

Table 7: Comparison with state of the art collaborative approaches on `Movielens`.

Strategy	P@10	R@10	nDCG@10
DB2vec SG 200 8	**0.0768**	**0.1777**	**0.2523**
MF	0.0173	0.0209	0.1423
PopRank	0.0397	0.0452	0.1598
BPRMF	0.0449	0.0751	0.1858
SLIM	0.0543	0.0988	0.2317
RankMF	0.0369	0.0459	0.1714

Table 8: Comparison with state of the art collaborative approaches on `LibraryThing`.

5. CONCLUSION

In this paper, we have presented an approach for learning low-dimensional real-valued representations of entities in RDF graphs, in a completely domain independent way. We have first converted the RDF graphs into a set of sequences using graph walks, which are then used to train neural language models. In the experimental section we have shown that a content-based RS relying on the similarity between items computed according to our latent features vectors, outdo the same kind of system but grounding on explicit features (e.g. types, categories,...) or features generated with the use of kernels, from both perspectives of accuracy and aggregate diversity. Our purely content-based system has been further compared to state of the arts collaborative approaches for rating prediction and item ranking, giving outstanding results on a dataset with a realistic sparsity degree.

As future work, we intend to introduce the features vectors deriving from the graph embeddings technique within a hybrid recommender system in order to get a fair comparison against state of the art hybrids approaches such as SPRank [10] and BRP-SSLIM [25]. In this perspective we could take advantage of the Factorization Machines [33], general predictor working with any features vector, that combine Support Vector Machines and factorization models. We aim to extend the evaluation to additional metrics, such as the individual diversity [44, 9], and to provide a deeper insight into cold-start users, i.e. users with a small interaction with the system for whom the information inference is difficult to draw and that generally benefit most of content "infusion".

6. REFERENCES

[1] Gediminas Adomavicius and YoungOk Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Trans. on Knowl. and Data Eng.*, 24(5):896–911, May 2012.

[2] Alejandro Bellogín, Iván Cantador, and Pablo Castells. A comparative study of heterogeneous item recommendations in social systems. *Inf. Sci.*, 221:142–169, February 2013.

[3] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data – The Story So Far. *International journal on semantic web and information systems*, 5(3):1–22, 2009.

[4] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, November 2002.

[5] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.

[6] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Narducci Fedelucio, and Giovanni Semeraro. Semantics-aware content-based recommender systems. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 119–159. Springer, 2nd edition, 2015.

[7] Gerben Klaas Dirk de Vries and Steven de Rooij. Substructure counting graph kernels for machine learning from rdf data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 35:71–84, 2015.

[8] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407, 1990.

[9] T. Di Noia, V. C. Ostuni, J. Rosati, P. Tomeo, and E. Di Sciascio. An analysis of users' propensity toward diversity in recommendations. In *ACM RecSys '14*, RecSys '14, pages 285–288. ACM, 2014.

[10] T. Di Noia, V. C. Ostuni, P. Tomeo, and E. Di Sciascio. Sprank: Semantic path-based ranking for top-n recommendations using linked open data. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2016.

[11] Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, and Davide Romito. Exploiting the web of data in model-based recommender systems. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 253–256, New York, NY, USA, 2012. ACM.

[12] Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, Davide Romito, and Markus Zanker. Linked open data to support content-based recommender systems. In *Proceedings of the 8th International Conference on Semantic Systems*, I-SEMANTICS '12, pages 1–8, New York, NY, USA, 2012. ACM.

[13] Mouzhi Ge, Carla Delgado-battenfeld, and Dietmar Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *In RecSys '10*, page 257, 2010.

[14] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1809–1818, New York, NY, USA, 2015. ACM.

[15] Z. S. Harris. *Mathematical Structures of Language*. Wiley, New York, NY, USA, 1968.

[16] Oktie Hassanzadeh and Mariano Consens. M.: Linked movie data base. In *In: Workshop on Linked Data on the Web*, 2009.

[17] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[18] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, SÃűren Auer, and Christian Bizer. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 2013.

[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[21] Cataldo Musto, Giovanni Semeraro, Marco De Gemmis, and Pasquale Lops. Word embedding techniques for content-based recommender systems: an empirical evaluation. In *RecSys Posters, ser. CEUR Workshop Proceedings, P. Castells, Ed*, volume 1441.

[22] Cataldo Musto, Giovanni Semeraro, Pasquale Lops, and Marco de Gemmis. *Random Indexing and Negative User Preferences for Enhancing Content-Based Recommender Systems*, pages 270–281. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[23] Cataldo Musto, Giovanni Semeraro, Pasquale Lops, and Marco de Gemmis. *Contextual eVSM: A Content-Based Context-Aware Recommendation Framework Based on Distributional Semantics*, pages 125–136. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[24] Xia Ning and George Karypis. SLIM: sparse linear methods for top-n recommender systems. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, pages 497–506, 2011.

[25] Xia Ning and George Karypis. Sparse linear methods with side information for top-n recommendations. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 155–162, New York, NY, USA, 2012. ACM.

[26] Tommaso Di Noia, Vito Claudio Ostuni, Jessica Rosati, Paolo Tomeo, Eugenio Di Sciascio, Roberto Mirizzi, and Claudio Bartolini. Building a relatedness graph from linked open data: A case study in the it domain. *Expert Systems with Applications*, 44:354 – 366, 2016.

[27] V. C. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In *ACM RecSys '13*, pages 85–92, 2013.

[28] Makbule Gulcin Ozsoy. From word embeddings to item recommendation. *arXiv preprint arXiv:1601.01356*, 2016.

[29] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, (Preprint):1–20, 2016.

[30] Heiko Paulheim and Johannes Fümkranz. Unsupervised generation of data mining features from linked open data. In *Proceedings of the 2nd international conference on web intelligence, mining and semantics*, page 31. ACM, 2012.

[31] Heiko Paulheim, Petar Ristoski, Evgeny Mitichkin, and Christian Bizer. Data mining with background knowledge from the web. *RapidMiner World*, 2014.

[32] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[33] Steffen Rendle. Factorization machines with libfm. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012.

[34] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009.

[35] Petar Ristoski, Christian Bizer, and Heiko Paulheim. Mining the web of linked data with rapidminer. *Web Semantics: Science, Services and Agents on the World Wide Web*, 35:142–151, 2015.

[36] Petar Ristoski, Eneldo Loza Mencía, and Heiko Paulheim. A hybrid multi-strategy recommender system using linked open data. In *Semantic Web Evaluation Challenge*, pages 150–156. Springer, 2014.

[37] Petar Ristoski and Heiko Paulheim. Rdf2vec: Rdf graph embeddings for data mining. In *International Semantic Web Conference (To Appear)*. Springer, 2016.

[38] Magnus Sahlgren. An introduction to random indexing. In *In Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005*, 2005.

[39] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. Adoption of the linked data best practices in different topical domains. In *International Semantic Web Conference*, pages 245–260. Springer, 2014.

[40] Giovanni Semeraro, Pasquale Lops, Pierpaolo Basile, and Marco de Gemmis. Knowledge infusion into content-based recommender systems. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 301–304, New York, NY, USA, 2009. ACM.

[41] Harald Steck. Evaluation of recommendations: Rating-prediction and ranking. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 213–220, New York, NY, USA, 2013. ACM.

[42] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[43] Markus Weimer, Alexandros Karatzoglou, and Alex Smola. Improving maximum margin matrix factorization. *Mach. Learn.*, 72(3):263–276, September 2008.

[44] Mi Zhang and Neil Hurley. Avoiding monotony: Improving the diversity of recommendation lists. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 123–130, New York, NY, USA, 2008. ACM.

ReDyAl: A Dynamic Recommendation Algorithm based on Linked Data*

Iacopo Vagliano*, Cristhian Figueroa*§, Oscar Rodríguez Rocha†,
Marco Torchiano*, Catherine Faron-Zucker‡, Maurizio Morisio*

*Dept. Control and Computer Engineering, Politecnico di Torino, Turin, Italy
{iacopo.vagliano, cristhian.figueroa, marco.torchiano, maurizio.morisio}@polito.it
§ Universidad del Cauca, Popayán, Colombia
† INRIA Sophia Antipolis Méditerranée, Sophia Antipolis, France
oscar.rodriguez-rocha@inria.fr
‡ Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, Sophia Antipolis, France
faron@i3s.unice.fr

ABSTRACT

The Web of Data is an interconnected global dataspace in which discovering resources related to a given resource and recommend relevant ones is still an open research area. This work describes a new recommendation algorithm based on structured data published on the Web (Linked Data). The algorithm exploits existing relationships between resources by dynamically analyzing both the categories to which they belong to and their explicit references to other resources. A user study conducted to evaluate the algorithm showed that our algorithm provides more novel recommendations than other state-of-the-art algorithms and keeps a satisfying prediction accuracy. The algorithm has been applied in a mobile application to recommend movies by relying on DBpedia (the Linked Data version of Wikipedia), although it could be applied to other datasets on the Web of Data.

Keywords

Recommender System, Linked Data, DBpedia, Semantic Web

1. INTRODUCTION

The Web is evolving from an information space for sharing textual documents into a medium for publishing structured data. Linked Data[1] is a set of best practices to publish and interlink data on the Web and it is the base of the Web of Data, an interconnected global dataspace where data providers publish their content publicly.

Due to the increase in the amount of structured data published on the Web through the principles of Linked Data, it is more likely to find resources that describe or represent real life concepts. The information provided by these resources may be used in many different domains. However, finding and recommending related resources is still an open research area [19]. The work presented in this paper holds on the results obtained from an our previous study and is its continuation[6]. The study stated that the problem of finding existing relationships between resources can be addressed by analyzing the categories they belong to, their explicit references to other resources and/or by combining both these approaches. The study also showed that many works aimed at resolving this problem by focusing on a specific application domain and dataset. In this paper, we address this issue and we focus on the following research questions: (i) *How can we design a recommendation algorithm that exploits existing relationships between resources on the Linked Data, is independent from the application domain and may be used on different datasets on the Web of Data?* (ii) *How can we design a recommendation algorithm that provides novel recommendations, i.e., recommendations of resources not previously known from the user, without affecting the prediction accuracy of the algorithm?*

We propose a new algorithm based on Linked Data which exploits existing relationships between resources in order to recommend related resources. It dynamically analyzes the categories they belong to and their explicit references to other resources, then combines the results. The algorithm has been applied to DBpedia[2], but it could as well be applied to other datasets on the Web of Data and it is not bound to any specific application domain.

We conducted a user study to comparatively evaluate its accuracy and novelty against three state-of-the-art algorithms, which showed that our algorithm provides a higher number of novel recommendations, while keeping a satisfying prediction accuracy. An implementation of our recommendation algorithm has been integrated into a mobile application suggesting movies based on DBpedia, which was developed in collaboration with Telecom Italia, the major network operator in Italy.

*The mobile application presented in Section 5 was done at the Joint Open Lab MobiLAB and was supported by a fellowship from TIM.

[1] `http://linkeddata.org`

[2] `http://dbpedia.org`

CBRecSys 2016, September 16, 2016, Boston, MA, USA.

The paper is organized as follows: Section 2 reviews related works; Section 3 presents our algorithm; Section 4 describes the evaluation method and provides the results; Section 5 shows the application of our algorithm for recommending movies; Section 6 provides conclusions.

2. RELATED WORK

This work began by conducting a systematic literature review [6] on the subject, which allowed us to lay the groundwork for this research. Such review listed the different approaches to exploit Linked Data in order to recommend resources. Some studies found, infer relationships between resources by taking into account the existing links between them in a dataset, and use these relationships to measure the semantic similarity of the resources. Such relationships can be direct links, paths, or shared topics between sets of items. The most important related works are summarized in the following.

Damljanovic et al. [4] recommended experts in an open innovation scenario. Their approach, named *HyProximity*, takes as input a description of a problem in natural language and extracts a set of relevant words that are linked with resources of DBpedia. Then it generates recommendations by combining two techniques. The first one consists in discovering resources related through hierarchical relationships, while the second one is based on traversal relationships, which connect resources without establishing a classification or hierarchy. By exploiting these two kinds of relationships, the approach identifies a set of direct or indirect topics related with potential experts to solve an innovation problem.

Passant [17] described *dbrec*, a recommender targeted for the music domain, which mainly relies on a distance measure named Linked Data Semantic Distance (LDSD). It takes into account the number of direct or indirect links between resources (related with the music domain) represented in DBpedia. Unlike HyProximity it does not distinguish between traversal and hierarchical links. Both Damljanovic et al. and Passant had to reduce the set of resources and links of the dataset to those belonging to a specific domain (innovation problems and music respectively), which involves a huge effort to manually define which resources or links should be considered.

Other works combine Linked Data based algorithms with other techniques of recommendation in order to improve the results. These techniques include collaborative filtering [10, 14, 16, 18], information aggregation [2, 9, 12] and statistical methods like Random Indexing (RI) [23], Vector Space Model (VSM) [1, 16], Latent Dirichlet Allocation (LDA) [11], implicit feedback [16] and structure-based statistical semantics [3]. De Graaff et al. [5] proposed a knowledge-based recommender system that derives the user interests from the users social media profile, which is enriched with information from DBpedia. Musto et al. [15] compared several techniques to automatically feed a graph-based recommender system with features extracted from Linked Data. However, these techniques usually require additional information from the user in order to produce accurate recommendations.

We propose a new recommendation algorithm, which is cross-domain and cross-dataset. It relies only on Linked Data and does not require to reduce the set of resources and links of the dataset to those belonging to a specific domain.

3. ReDyAl

ReDyAl is a recommendation algorithm which takes into account the different types of relationships between the data published according to the Linked Data principles. It aims at discovering related resources from datasets that may contain either *well-linked* resources as well as *poorly-linked* resources. A resource is said to be well-linked if it has a number of links higher than the average number of links in the dataset; otherwise it is poorly-linked. The algorithm is able to dynamically adapt its behavior in order to find a set of candidate resources to be recommended, relying on the implicit knowledge contained in the Linked Data relationships.

3.1 Principles

Any dataset on the Web of Data may be seen as a tuple *(R, T, L)* composed by resources (R), categories (T), and relationships (L). Categories denote types, concepts or classes. Resources are instances of concepts; they are Web resources or real world resources identified by a URI. Relationships are also known as links or properties; they are the links connecting resources or categories along the whole dataset graph. Categories often are hierarchically organized. For example, DBpedia provides information about hierarchical relationships in three different classification schemata: Wikipedia Categories, YAGO[3] [24] classes, and WordNet Synsets[4]. Relationships can be of three types:

Resource-Resource (*R-R*) These are the traversal relationships between resources, i.e. the links between resources that do not refer to hierarchical classifications.

Resource-Category (*R-T*) These are relationships between a resource and a category. They can be represented by the RDF[5] property `rdf:type` or the `dcterms:subject` property from the Dublin Core vocabulary[6].

Category-Category (*T-T*) These are hierarchical relationships between categories within a hyponymy structure (a category tree). They can be represented by using the RDFS[7] property `rdfs:subClassOf` or the SKOS[8] properties `skos:broader` (isSubCategoryOf) and `skos:narrower` (isSuperCategoryOf).

Considering this model of a dataset, ReDyAl consists in three stages:

1. The first stage discovers resources by analyzing the links between the given initial resource and other resources. Only R-R relationships are considered at this stage, although they can be indirect, i.e. they can connect two resources through a third one.

[3] `http://www.mpi-inf.mpg.de/yago-naga/yago/`
[4] `https://wordnet.princeton.edu`
[5] `http://www.w3.org/TR/rdf11-concepts/`
[6] `http://dublincore.org/documents/dcmi-terms/`
[7] `http://www.w3.org/TR/rdf-schema/`
[8] `https://www.w3.org/TR/skos-reference`

2. The second stage analyzes the categorization of the given initial resource and discovers similar resources located in the same categories. It finds indirect relationships between resources through direct R-T and T-T relationships. It is possible to specify to the algorithm which specific R-T and T-T relationships to consider in this step: the choice for R-T relationships is between `dcterms:subject` or `rdf:type`, while `skos:broader` and `skos:narrower` or `rdfs:subClassOf` are acceptable T-T relationships.

3. The last stage intersects the results of both the previous stages and ranks them by giving priority to those found in the first stage. The algorithm computes the similarity of the initial resource with respect to any of the discovered resources, based on a similarity function which combines the Linked Data Semantic Distance (LDSD) [17] and HyProximity distance [4], opportunely adapted and generalized.

The algorithm can be applied to any dataset on the Web of Data. In the first step, it relies only on R-R relationships: any relationship of this kind may be used, independently of the data stored on the dataset. In the second step, the algorithm can be configured to use the `dcterms:subject` or `rdf:type` properties, which are R-T relationships. DBpedia uses both to enable different categorizations; for example to rely on the Wikipedia categories, it is necessary to set `dcterms:subject` as R-T relationship and `skos:broader` and `skos:narrower` as T-T relationships. Any other dataset uses at least `rdf:type` to indicate the class which a resource is instance of. Thus, `rdf:type` can be used to find resources in the same class and then `rdfs:subClassOf` can be used to retrieve more general classes (or `skos:broader` and `skos:narrower`, if the categories are organized through SKOS properties).

The algorithm is independent on the application domain because it relies only on R-R, R-T or T-T links. If in the dataset on which the algorithm is applied there are relationships among resources in different domains the algorithm may generate cross-domain recommendations. For example, DBpedia is a general dataset which represents resources of different kind and there may be a relationship between a song and a city because the song was recorded in that city, or because is about the city. Alternatively, there may be a link between a song and a movie because the song was part of the soundtrack of the movie. Thus, a city or a movie may be recommended starting from a song. Also R-T links may generates cross-domain recommendations if resources which belong to different domains are included into the same category.

3.2 Reducing the search space

Additionally, the algorithm may be configured with a set of forbidden links in order to restrict the kind of links the algorithm should consider. This is useful to prevent the algorithm to obtain resources over links pointing to empty nodes (i.e. resources without a URI), literals that are used to identify values such as numbers and dates, and other nodes that are not desired for the recommendation. In other words, it is a way to limit the results of the algorithm. For example the DBpedia resource `dbr:Turin` contains the link `dbpprop:populationTotal` that points to the integer value 911823: we can confgure this link as forbidden link since it does not point to a resource which can be recommended. This is also useful to increase the performance of the algorithm because limiting the number of results decreases the ranking time. All the links which are not explicitly specified as forbidden are allowed links and define a domain of interest. This may be useful when the algorithm is applied to a generic dataset as DBpedia. This dataset contains millions of links between resources, and if a developer is creating an application in the music domain then he/she may be interested only in resources of that domain, so he/she may want to consider only links pointing to those resources i.e., a set of allowed links. In fact the algorithm is cross-domain, thus it may recommend a city or a movie starting from a song, as we have already explained. While this may be an advantage in some applications, it may be confusing in others, especially if not properly explained to the user. To limit the recommendations to specific categories of resources (for example to consider only tracks and artists) it is sufficient to "allow" only the relationships which point to these kinds of resources, i.e. which have such desired category as range.

3.3 Parameter Settings

ReDyAl receives as input an initial resource by specifying its corresponding URI ($inURI$), and three values ($minT$, $minC$, $maxDistance$) for configuring its execution. The selection of $minT$ and $minC$ is arbitrary and depends on the dataset and the convenience of the user who is setting up the algorithm. $minT$ is the minimum number of links (input and output links involving the initial resource) necessary to consider a resource as well-linked. The proper value of $minT$ depends on the dataset: if it contains resources with a high number of links between them it is expected to be higher, while if the resources have only few links it should be set to a lower value. However, this parameter impacts on the algorithm: if the initial resource is well-linked, traversal interlinking has a higher priority in the generation of candidate resources, otherwise the algorithm gives priority to the hierarchical relationships. For example, a user may consider the use of the hierarchical algorithms only if the resources are connected with less than 10 links by setting $minT$ to 10. In a similar way, the user may arbitrary fix the value of $minC$, which is the minimum number of candidate resources that the algorithm is expected to generate, i.e. the number of candidate resources the user is expecting.

The value of $maxDistance$ limits the distance (i.e. the number of hierarchical levels) that the algorithm considers in a category tree. $maxDistance$ may be defined manually; this is particularly useful when there are not enough candidate resources from the categories found at a certain distance (i.e. the number of candidate resources retrieved is lower than $minC$). In this case, the algorithm increases the distances in order to find more resources and if the $maxDistance$ value is reached with less than $minC$ candidate resources, the algorithm ranks only the candidate resources found until that moment. Additionally, the algorithm may receive a list of forbidden links (FL) to avoid searching for candidate resources over a predefined list of undesired links.

3.4 Algorithm

Algorithm 1 ReDyAl algorithm

Require: $inURI$, $minT$, $minC$, FL, $maxDistance$,
Ensure: A set of candidate resources CR

```
1: L_in = readAllowedLinks(inURI, FL)
2: if |L_in| ≥ minT then
3:   for all l_k ∈ L_in do
4:     DRl_k = getDirectResources(l_k)
5:     IRl_k = getIndirectResources(l_k)
6:     Add DRl_k to CR_tr
7:     Add IRl_k to CR_tr
8:   end for
9:   if |CR_tr| ≥ minC then
10:    return CR_tr
11:  else
12:    currentDistance = 1
13:    Gc = createCategoryGraph(inURI, currentDistance)
14:    while currentDistance ≤ maxDistance do
15:      CR_hi = getCandidateResources(Gc)
16:      if |CR_hi| ≥ minC then
17:        Add CR_tr and CR_hi to CR
18:        return CR_hi
19:      end if
20:      increase currentDistance
21:      updateCategoryGraph(currentDistance)
22:    end while
23:    Add CR_tr and CR_hi to CR
24:  end if
25: end if
26: return CR
```

ReDyAl (Algorithm 1) starts by retrieving a list of allowed links from the initial resource. Allowed links are those that are not specified as forbidden (FL) or that are explicitly defined in the initial resource. If there is a considerable number of allowed links (more than $minT$, i.e., the initial resource is well-linked) the algorithm obtains a set of candidate resources located through direct (DRl_k) or indirect traversal links (IRl_k), starting from the links explicitly defined in the initial resource (Lines 1-8). A resource is indirectly linked to the initial resource if it is linked through another resource. A resource directly linked is located at traversal distance 1 from the initial resource, while a resource indirectly linked is located at traversal distance 2 from the initial resource. With regards to the traversal links, a maximum distance of 2 is considered because for distances higher than 2 (i.e 1 direct heap plus 1 indirect heap) the number of retrieved resources is dramatically increased, therefore increasing also the number of resources that are not relevant or related with the initial resource.

Next, if the current number of candidate resources generated (CR_{tr}) is greater than or equal to $minC$, the algorithm terminates returning the results (Lines 9-10). Otherwise, the algorithm generates a category graph (Gc) with categories of the first distance and applies iterative updates over the category graph over n distances from the initial resource, obtaining broader categories (i.e. more generic categories that are located in a higher level in a classification) until at least one of two following conditions is fulfilled: the number of candidate resources is sufficient ($|CR| > minC$), or the maximum distance is reached ($currentDistance > maxDistance$). At each iteration, candidate resources (CR_{hi}) are extracted from the broader categories of maximum distance (Lines 14-23). In any case, the algorithm combines these results with the results obtained in Lines 3-8 (adding CR_{tr} and CR_{hi} to CR). Finally, the set of candidate results is returned (Line 23).

3.5 Ranking of the recommended resources

The final operation is ranking the sets of candidate resources. The ranking process receives as input the candidate resources retrieved by the ReDyAl algorithm and ranks them according to their degree of similarity with the initial resource. This similarity is computed based on a combination of two distance measures: LDSD and HyProximity.

The LDSD distance, initially proposed by Passant [17], is based on the number of indirect and direct links between two concepts. In this measure, the similarity of two resources (r_1, r_2) is measured by combining four properties: the input/output direct links or the input/output indirect links between them. Equation 1 presents the basic form of the $LDSD$ distance. Cd_{out} is the number of direct output links (from c_1 to c_2), Cd_{in} is the number of direct input links, Ci_{in} is the number of indirect input links, and Ci_{out} is the number of indirect output links. The implementation developed by Passant is limited to links from a specific domain, while the LDSD function implemented in ReDyAl takes into account all the concepts of the dataset unless forbidden links are specified.

$$LDSD(c_1, c_2) = \frac{1}{1 + Cd_{out} + Cd_{in} + Ci_{out} + Ci_{in}} \quad (1)$$

HyProximity is a similarity measure defined by Stankovic et al. [4], which can be used to calculate both traversal and hierarchical similarities. The measure in its general form is shown in Equation 2 as the inverted distance between two concepts, balanced with a pondering function. In this equation $d(r_1, r_2)$ is the distance function between the resources r_1 and r_2, while $p(r_1, r_2)$ is the pondering function, which is used to weight different distances. Based on the structural relationships (hierarchical and traversal), different distance and pondering functions may be used to calculate the HyProximity similarity. ReDyAl reuses the HyProximity hierarchical measure, which is the quotient of a pondering function (p) and a distance (d). The distance was calculated using $maxDistance$ such that: $d(ir, r_i) = maxDistance$, where ir is the initial resource and r_i is a candidate resources generated by the recommendation algorithm. The pondering function was calculated with an adaptation of the informational content function (Equation 3) defined by Seco et al. [21]. In this equation $hypo(C)$ is the number of descendants of the category C and $|C|$ is the total number of categories in the category graph. This function was selected because it minimizes the complexity of calculation of the informational content, compared to other functions that employ an external corpus [8]. Nonetheless, in ReDyAl, this measure is not limited to a specific property, and optionally can be configured to support a set of forbidden links.

$$hyP(r_1, r_2) = \frac{p(r_1, r_2)}{d(r_1, r_2)} \quad (2)$$

$$p(C) = 1 - \frac{\log(hypo(C) + 1)}{log(|C|)} \quad (3)$$

$$Hybrid_{sim} = (1 - LDSD)\alpha + (hyP(r_1, r_2))\beta \quad (4)$$

Finally, the measure that combines LDSD and HyProximity used by ReDyAl is defined in Equation 4, where α and β may be set according to the convenience of the user: α is the weight for the traversal algorithm and β is the weight for the hierarchical algorithm. In this way, resources are ranked in descending order, arranged from the largest to the smallest value of $Hybrid_{sim}$.

4. USER EVALUATION

We comparatively evaluated the prediction accuracy and the novelty of the resources recommended with ReDyAl with respect to three state-of-the-art recommendation algorithms relying exclusively on Linked Data to produce recommendations: dbrec [17], HyProximity traversal and HyProximity hierarchical [4]. This evaluation aimed to answer the following questions: (RQ1) *Which of the considered algorithms is more accurate?* (RQ2) *Which of the considered algorithms provides the highest number of novel recommendations?*

We decided to rely on a user study because we were interested in evaluating the novelty of proposed recommendations over the accuracy. Since we cannot expect that users rated all the items they already know, a user study can measure novelty more precisely than an offline study. On the other side, user studies are more expensive to conduct than an offline studies, for this reason we focus on recommendation algorithms based only on Linked Data and we did not consider algorithm exploiting traditional techniques, or combining Linked Data with traditional techniques. We plan to conduct other experiments to compare our method with other techniques and investigate on the effectiveness of our approach combined with traditional techniques.

Although our algorithm is not bound to any particular dataset, we applied it to DBpedia because it is a general dataset that offers the possibility to evaluate the results in a number of scenarios. DBpedia is one of the biggest datasets in the Web of Data and the most interlinked [20]. Furthermore, it is frequently updated and continuously grows.

4.1 Experiment

A user study was conducted involving 109 participants. The participants were mainly students of Politecnico di Torino (Italy) and University of Cauca (Colombia) enrolled in IT courses. The average age of the participants was 24 years old and they were 91 males, 14 females, and 4 of them did not provide any information about their sex. Although the proposed algorithm is not bound to any particular domain, this evaluation focused on movies because we aimed at applying our algorithm in the mobile application presented in Section 5 (which suggest movies) and in this domain a quite large amount of data is available on DBpedia. Additionally, it was easier to find participants, since no specific skills are required to express an opinion about movies. The algorithms were compared within subjects [22] since each participant evaluated recommendations from different algorithms, as it is explained in the following.

The evaluation was conducted as follows. A list of 20 recommendations generated from a given initial movie was presented to the participants. For each recommendation two questions were asked: *(Q1) Did you already know this recommendation? Possible answers were: yes, yes but I haven't seen it (if it is a movie) and no. (Q2) Is it related to the movie you have chosen? Possible answers were: I strongly agree, I agree, I don't know, I disagree, I strongly disagree. Each answer was assigned respectively a score from 5 to 1.*

We developed a website[9] to collect the answers from the participants. The participants were able to choose an initial movie from a list of 45 movies selected from the IMDB top 250 list[10]. The first 50 movies were considered and 5 movies were excluded because they were not available in DBpedia. Choosing these movies ensured participants to know them, but was also a limitation: the corresponding DBpedia resources are very well-linked, thus we could not properly evaluate the algorithm on poorly linked initial movies. The movies were presented to the user in a random order to avoid having most of the participants evaluating recommendations for the same initial movies (e.g. the first in the lists). When a participant selected an initial movie the tool provided the corresponding list of recommendations with the questions mentioned above. The recommendations were presented in a randomized order. Each participant was able to evaluate recommendations from as many initial movies as he wanted, but he had to answer the questions for all the recommendations, i.e. was not possible to answer only to part of the questions for the initial movie chosen. As a result, the recommendations of the lists for 40 out of 45 initial movies were evaluated by at least one participant and each movie was evaluated by an average of 6.18 participants. The dataset with the initial movies and the lists of recommendations is available online[11].

Each list of 20 recommendations was pre-computed. In particular, recommendations were generated for each of the 45 initial movies with each of the four different algorithms. Then, the recommendations generated by each algorithm were merged in a list of 20 recommendations to be shown to the participants. To do this, we generated a list of 40 recommendations by selecting the first 10 pre-computed recommendations for each algorithm and we ordered them by the similarity computed by each algorithms, since each algorithm ranks its recommendations by using its semantic similarity function with values between 0 and 1. Then we eliminated eventual duplicates, since the same recommendation could be provided by more than one algorithm. The final list was obtained considering the first 20 recommendations of the merged list.

With regard to the questions stated at the beginning of this section, to answer RQ1, the Root Mean Squared Error (RMSE) [22] was computed, and to answer RQ2 the ratio between the number of evaluations was computed in which the recommended item was not known by the participants and the total number of evaluations. For the RMSE measure, scores given by the participants when answering to

[9] `http://natasha.polito.it/RSEvaluation/`
[10] `http://www.imdb.com/chart/top`
[11] `http://natasha.polito.it/RSEvaluation/faces/resultsdownload.xhtml`

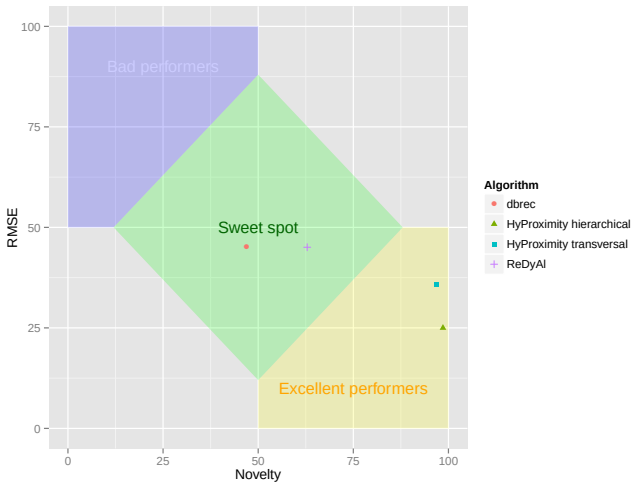


Figure 1: Prediction accuracy and novelty of the algorithms evaluated

Q2 were considered as reference and were normalized in the interval [0, 1], and these scores were compared with the similarities computed by each algorithms, since each algorithm ranks its recommendations by using its semantic similarity function.

4.2 Results

The results of the evaluation are summarized in Figure 1, which compares the algorithms with respect to their RMSE and novelty. The "sweet spot" area represents the conditions in which an algorithm has a good trade-off between novelty and prediction accuracy. In effect, presenting a high number of recommendations not known to the user is not necessarily good because it may prevent him to assess the quality of the recommendations: for example having in the provided recommendation a movie which he has seen and which he liked may increase the trust of the user in the RS.

Regarding RQ1, HyProximity accounts for the lowest RMSE measures (with 25% and about 36% for the hierarchical and traversal versions respectively), but these results are less significant due to the low number of answers to Q2 for these algorithms (this means that the RMSE was computed over a low number of recommendations). For both ReDyAl and dbrec the RMSE is roughly 45%. Concerning RQ2, the two versions of HyProximity account for the highest values (hierarchical roughly 99%, while traversal about 97%). However, such a high rate of novel recommendations may confuse the user and prevent him to judge recommendations, as we have already explained. ReDyAl has a larger rate of novel recommendations than dbrec. These two algorithms account respectively for about 60% and 45%.

The recommendations generated by HyProximity in both traversal and hierarchical version collected a low number of answers to Q2 because most of the recommendations generated by these algorithms were unknown as illustrated in Table 1. Consequently the RMSE was computed over a low number of recommendations. Thus, the results of these two algorithms related to RQ1 are less definitive than for the others, since for measuring the prediction accuracy only the evaluations for which the answer to Q1 was either "yes" or "yes but I haven't seen it (if it is a movie)" were considered.

We computed the Fleiss' kappa [7] measure for assessing the agreement of the participants in answering Q2. We considered the recommendations and in particular we considered as different the same recommendation when related to a different initial movie (i.e. when appearing in different lists of recommendations). We excluded recommendations not evaluated or evaluated by only one participant. The Fleiss' kappa is 0.79; according to Landis and Koch [13], this corresponds to a substantial agreement.

In conclusion, Figure 1 illustrates that ReDyAl and dbrec provides a good trade-off between prediction accuracy and novelty (sweet spot area), although ReDyAl performs better in novelty. HyProximity hierarchical and HyProximity traversal seem to be excellent performers since the RMSE is low and the novelty is high, but the RMSE was computed on few evaluations. An additional analysis of these two algorithms is needed to verify if the user can benefit from such a high novelty and if novel recommendations are relevant. In addition, further investigation is needed on poorly-linked resources, since the choice of the initial movies focused on selecting well known movies to make easier the evaluation from participants, but the related resources were well-linked. On poorly-linked resources we expect ReDyAl and Hyproximity hierarchical keeping good recommendations since they can rely on categories, while dbrec and HyProximity traversal are likely to provide much less recommendations since they rely on direct links between resources.

5. MOBILE MOVIE RECOMMENDATIONS

An implementation of ReDyAl has been integrated into a mobile application developed in collaboration with Telecom Italia (the major network operator in Italy). This application recommends movies based on DBpedia: when the user enters the title of a movie, the application provides the Wikipedia categories to which the initial movie is related to. In this way, the user may focus on a specific scope and can receive recommendations of related resources for any category. In addition, it is possible to view any recommendation to obtain additional information.

Our algorithm can provide cross-domain recommendations because it is independent on the domain and is applied on DBpedia, which is a general dataset. Thus, the recommended resources can be movies but also other relevant entities such as actors, directors, places of recording, books on which the movie is inspired, etc. Other advantages of using DBpedia as dataset are the high number of resources that it represents, the variety of domains addressed and the continuous update and growth, since it is extracted from Wikipedia.

For example, given *The Matrix* as initial movie the categories which it belongs to are presented. The user may be more interested in martial arts, post-apocalyptic movies or he may prefer to consider all the movies from American directors, thus he can choose a category accordingly. By selecting Post-apocalyptic films, a number of resources are recommended. For each recommendation it is possible to open

Algorithm	Yes	Yes but I haven't seen it	No
ReDyAl	27.95	9.17	62.88
dbrec	41.10	11.95	46.95
HyProximity hierarchical	1.08	0.36	98.56
HyProximity traversal	1.32	1.89	96.79

Table 1: Percentage of answers for Q1 by algorithm

a detailed view, which contains three tabs: the first contains a brief textual description, the second presents a graph view of the resource in order to show the main properties and the third summarizes the main information in a tabular form. The graph view is illustrated in Figure 2. The graph is paginated and few properties per page are presented in order to avoid information overload, since the resource can have a very high number of properties. This view can be useful also to explain the recommendation: for instance the user can understand that the recommended resource has the same director or the same main actor as the initial movie. The graph view is based on DBpedia Mobile Explorer [25], a Linked Data visualization framework for the mobile environment, which enables the application to hide the underlying complexity of the Linked Data to the users by processing the resources to be presented received from DBpedia.

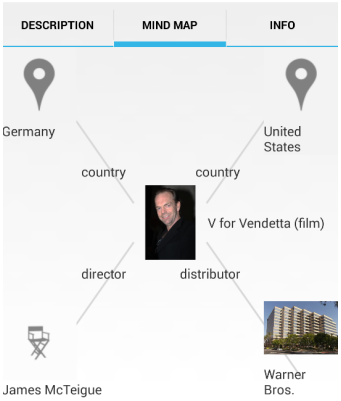


Figure 2: The graph view of *V for Vendetta*

The application is based on a client-server architecture and the main modules are DBpedia, a RESTful recommender service[12] which exposes our algorithm, and the mobile user interface. The main flow of interactions is represented in Figure 3. The mobile application asks for recommendations specifying an initial resource and optionally a scope such as a Wikipedia category (1). The recommender service answers with a list of scopes if no scope was provided or with a list of recommendations in the scope specified, otherwise (2). The recommender service relies on DBpedia to provide recommendations (3, 4) and the mobile application retrieves the resources to be visualized from the dataset (5, 6). The recommender service is developed in Java, while the client is an Android mobile application. The two modules use JSON as data-interchange format, while the mobile application retrieves resources from DBpedia serialized in JSON-LD[13]. The mobile application is going to be published on Google Play, but the Android Package (APK) of the first version is already available on the Web[14].

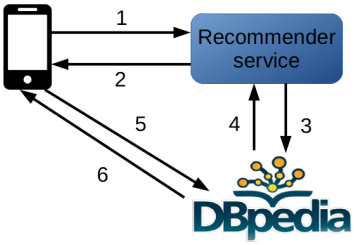


Figure 3: The interactions between the main modules of the application

6. CONCLUSIONS AND FUTURE WORK

We presented ReDyAl which is a hybrid algorithm that dynamically uses both the traversal and hierarchical approach for discovering resources. It is independent from the application domain and, although we applied it to DBpedia, it could be easily adapted to other dataset in the Web of Data. It relies only on Linked Data and does not require to reduce the set of resources and links of the dataset to those belonging to a specific domain.

We evaluated and compared our algorithm against three state-of-the-art algorithms by conducting a user study and we also showed a practical application of the algorithm by presenting a mobile application that provides movie recommendations relying on DBpedia. Although the algorithm could be applied to other datasets in the Web of Data, we selected DBpedia because it is a general dataset, thus cross-domain recommendations were possible. In addition, there is a high number of resources represented, a variety of domains addressed and it is continuously updated, since it is extracted from Wikipedia. The user study demonstrated that ReDyAl improves in the novelty of the results discovered, although the accuracy of the algorithm is not the highest (due to its inherent complexity). Although ReDyAl is not bound to any particular domain, the study focused on movies as for this domain there is a quite large amount of data available on DBpedia and participants were not required to have specific skills.

Future work includes studying the relevance under different domains and improving the accuracy of ReDyAl while maintaining its novelty. We plan to conduct other studies to compare it with traditional techniques and with approaches which combine Linked Data with traditional techniques. We are also working on combining ReDyAl with collaborative filtering techniques in order to take user preferences into account while providing recommendations.

[12] http://natasha.polito.it/LDRecommenderWeb/
[13] http://json-ld.org/
[14] https://www.dropbox.com/sh/0q8d2mcbko9e2oj/AAASh-YHGzOMmG_Z8hH6mfWOa?dl=0

7. REFERENCES

[1] S. Baumann, R. Schirru, and B. Streit. Towards a storytelling approach for novel artist recommendations. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6817 LNCS, pages 1–15, 2011.

[2] I. Cantador, I. Konstas, and J. M. Jose. Categorising social tags to improve folksonomy-based recommendations. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(1):1–15, Mar. 2011.

[3] G. Cheng, S. Gong, and Y. Qu. An Empirical Study of Vocabulary Relatedness and Its Application to Recommender Systems. In *10th International Conference on The Semantic Web - Volume Part I*, pages 98–113. Springer, 2011.

[4] D. Damljanovic, M. Stankovic, and P. Laublet. Linked data-based concept recommendation: Comparison of different methods in open innovation scenario. 7295:24–38, 2012.

[5] V. de Graaff, A. van de Venis, M. van Keulen, and R. A. de By. Generic knowledge-based analysis of social media for recommendations. In *CBRecSys 2015: New trends on content-based recommender systems*. CEUR-WS.org, Sept 2015.

[6] C. Figueroa, I. Vagliano, O. Rodríguez Rocha, and M. Morisio. A systematic literature review of linked data-based recommender systems. *Concurrency and Computation: Practice and Experience*, 2015.

[7] J. Fleiss et al. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.

[8] M. Hadj Taieb, M. Ben Aouicha, M. Tmar, and A. Hamadou. New information content metric and nominalization relation for a new wordnet-based method to measure the semantic relatedness. In *Cybernetic Intelligent Systems (CIS), 2011 IEEE 10th International Conference on*, pages 51–58, Sept 2011.

[9] H. Hamdan. Experiments with DBpedia , WordNet and SentiWordNet as re- sources for sentiment analysis in micro-blogging. In *Seventh International Workshop on Semantic Evaluation (SemEval 2013) - Second Joint Conference on Lexical and Computational Semantics*, volume 2, pages 455–459, Atlanta, Georgia, 2013.

[10] Y. Kabutoya, R. Sumi, T. Iwata, and T. T. Uchiyama. A topic model for recommending movies via linked open data. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, pages 625–630. IEEE, Dec. 2012.

[11] H. Khrouf and R. Troncy. Hybrid event recommendation using linked data and user diversity. In *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13*, RecSys '13, pages 185–192. ACM Press, 2013.

[12] K. Kitaya, H.-H. Huang, and K. Kawagoe. Music Curator Recommendations Using Linked Data. In *Second International Conference on the Innovative Computing Technology (INTECH 2012)*, pages 337–339. IEEE, Sept. 2012.

[13] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.

[14] E. Mannens, S. Coppens, T. De Pessemier, H. Dacquin, D. Van Deursen, R. De Sutter, and R. Van de Walle. Automatic news recommendations via aggregated profiling. *Multimedia Tools and Applications*, 63(2):407–425, 2013.

[15] C. Musto, P. Basile, M. de Gemmis, P. Lops, G. Semeraro, and S. Rutigliano. Automatic selection of linked open data features in graph-based recommender systems. In *CBRecSys 2015: New trends on content-based recommender systems*, pages 10–13. CEUR-WS.org, Sept 2015.

[16] V. C. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi. Top-N recommendations from implicit feedback leveraging linked open data. In *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13*, RecSys '13, pages 85–92. ACM Press, 2013.

[17] A. Passant. dbrec - Music Recommendations Using DBpedia. In *The Semantic Web - ISWC 2010*, pages 209–224. Springer Berlin Heidelberg, 2010.

[18] L. Peska and P. Vojtas. Enhancing Recommender System with Linked Open Data. In *10th International Conference on Flexible Query Answering Systems (FQAS 2013)*, pages 483–494, Granada, Spain, 2013. Springer Berlin / Heidelberg.

[19] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 1–35. Springer US, 2011.

[20] M. Schmachtenberg, C. Bizer, and H. Paulheim. Adoption of the linked data best practices in different topical domains. In *The Semantic Web - ISWC 2014*, volume 8796 of *Lecture Notes in Computer Science*, pages 245–260. Springer International Publishing, 2014.

[21] N. Seco, T. Veale, and J. Hayes. An Intrinsic Information Content Metric for Semantic Similarity in WordNet. In *European Conference on Artificial Intelligence*, pages 1089–1090, 2004.

[22] G. Shani and A. Gunawardana. Evaluating recommendation systems. *Recommender Systems Handbook*, pages 257–297, 2011.

[23] M. Stankovic, W. Breitfuss, and P. Laublet. Discovering Relevant Topics Using DBPedia: Providing Non-obvious Recommendations. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, pages 219–222. IEEE, Aug. 2011.

[24] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM.

[25] I. Vagliano, M. Marengo, and M. Morisio. DBpedia Mobile Explorer. In *1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 181–185, Sept 2015.

Quote Recommendation for Dialogs and Writings

Yeonchan Ahn*, Hanbit Lee*, Heesik Jeon**, Seungdo Ha* and Sang-goo Lee*
*School of Computer Science and Engineering, Seoul National University, Seoul, Korea
**Artificial Intelligence Team, Samsung Electronics Co., Ltd., Seoul, Korea
*{skcheon, acha21, seungtto, sglee}@europa.snu.ac.kr, **heesik.jeon@samsung.com

ABSTRACT
Citing proverbs and (famous) statements of other people can provide support, shed new perspective, and/or add humor to one's arguments in writings or dialogs. Recommending quote for dialog or writing can be done by considering the various features of the current text called context. We present five new approaches to quote recommendation: 1) methods to adjust the matching granularity for better context matching, 2) random forest based approach that utilizes word discrimination, 3) convolutional neural network based approach that captures important local semantic features, 4) recurrent neural network based approach that reflects the ordering of sentences and words in the context, and 5) rank aggregation of these algorithms for maximum performance. We adopt as baseline state-of-the-arts in citation recommendation and quote recommendation. Experiments show that our rank aggregation method outperforms the best baseline by up to 46.7%. As candidate quotes, we use famous proverbs and famous statement of other person in dialogs and writings. The quotes and their contexts were extracted from Twitter, Project Gutenberg, and Web blog corpus.

CCS Concepts
• Information systems ~ Recommender systems • Natural language processing.

Keywords
Quote recommendation; Context matching; Random forest; Convolutional Neural Network; Recurrent Neural Network; Rank aggregation

1. INTRODUCTION
Citing proverbs and (famous) statements of other people is an important part in conversation and writing. Such quotes or quotations can provide support, shed new perspective, and/or add humor to one's arguments. However, it is not easy for a person to find from a large number of quotes an appropriate one for a given context since the words in quote are usually metaphorical.

Quote recommendation in writing has been introduced in Tan, et al. [7]. Quote recommendation is a task of recommending a ranked list of quotes which are relevant to the current body of text which we call *context*. We separate *context* into *pre-context* and *post-context*, which refer to texts that appear before and after a quote within certain fixed length respectively. For dialogs, unlike for writings, we only use pre-contexts because post-contexts are usually unavailable for on-the-fly recommendation of quotes during a conversation in real world applications. We define *query* as a context for which the user desires a list of recommended quotes. Figure 1 shows an example of quote usage in our Twitter dataset. In this example, the block of text that appears before the quote '*Strike while the iron is hot*' is the pre-context.

CBRecSys 2016, September 16, 2016, Boston, MA, USA.

Figure 1 An example of quote usage in Twitter thread

On investigating our collected datasets, we found that various features of context, such as keywords, topic, n-grams, latent semantics, etc., can be exploited in the recommendation. For example, word matching-based algorithm such as ranking with cosine similarity between query and context of quote was able to find the correct quote in Figure 1, since many contexts of the same quote in training dataset mention the keywords such as *casino* and *luck* but the others do not. Also, some of the quotes are closely related to specific situations, topic or semantics behind query not to only keywords.

In this paper, we present five new approaches for quote recommendation based on observations in our datasets: 1) methods to adjust matching granularity for better context matching, 2) Random Forest (RF) based approach that utilizes word discrimination, 3) convolutional neural network (CNN) based approach that captures important local semantic features, 4) recurrent neural network (RNN) based approach that reflects the ordering of sentences and words in the context, and 5) rank aggregation of these algorithms for maximum performance. As baseline, we adopt previous works on citation recommendation [1, 3] and quote recommendation [7]. Experiments show that the proposed approaches significantly outperform baseline methods in real world datasets.

2. RELATED WORKS
Quote recommendation can be viewed as task of searching or recommending short texts which are appropriate to given current writing or dialog context. Most related works are citation recommendation for academic articles [1, 3], which recommends relevant reference articles for academic writing. For citation recommendation, rich information on paper such as title, abstract, full text and venue can be exploited. In contrast, in quote recommendation such rich information is not available. This makes quote recommendation more challenging. Tan, et al. [7] present a method for recommending quote for the first time. They apply learning-to-rank approach with several features which is quote-context, context-context (or context-query), and quote feature. In their experiments, they show that the algorithm heavily depends on context-context feature. However, we argue that enough exploration on the context-context features is not

conducted. For this, we focus on how to mine the semantics on contexts of quote for recommending quote.

3. APPROCHES

In this section, we describe four approaches and our rank aggregation method which combines the four approaches for quote recommendation.

3.1 Matching Granularity Adjustment

In this section we discuss methods to deal with the contexts of quotes when measuring relevance between query and a set of contexts of quotes, which we call matching granularity adjustment. As usage of words or words themselves in the quote are different from that in context, the state-of-the-arts in quote/citation recommendation [1, 3, 7] measures the relevance between query and contexts of a quote. More specifically, all of them attempt to examine individual context of a quote to the query. A drawback of this approach is that it suffers from sparsity problem that words in query do not match the individual context of the correct quote. In order to alleviate this sparsity problem, we propose methods to adjust the matching unit of contexts to the given query. We believe that more semantics can be exploited if the contexts of a quote are treated collectively.

Firstly, we propose a method called context clustering, which group the context by context cluster which represent (latent) topic. In the collected dataset, we observed that there exist a number of quotes that can be used in different topic. For example, the quote '*All work and no play makes jack a dull boy*' can be used in very different situations such as '*overworking in workplace*' or '*educating children*'. Thus when dealing with query about specific topic, we need to consider the contexts related to it among different topics of quote. In the context clustering, we first clusters contexts of each quote. And we exploit the context clusters to measure the relevance of a quote. For context clustering, we adopt affinity propagation clustering algorithm, which is known to perform better than others in short text clustering [6]. Based on context clustering, we propose a scoring function given query q:

$$sim_{\text{cmax}}(q,t) = \max(sim(q,\ CC_t^{(j)}))$$

where $CC_t^{(j)}$ is concatenated text in jth context cluster of quote t and sim is cosine similarity with their TF-IDF vector representation.

In order to solve the sparsity problem, we present another method called context lumping to adjust the matching granularity. In context lumping, we simply concatenate all the context of each quote and make it a matching unit to the query. Then the lumped context of quote is compared to query with cosine similarity with TF-IDF vector representation. In both of context clustering and lumping, quotes are sorted by the proposed similarities in descending order respectively.

3.2 Random Forest

In the collected dataset, we observe that some simple rules such as checking whether the given context contains certain words are reliable cursors to its correct label. For example, in Twitter dataset, given that a context contains the keywords *invite*, *join*, *come over* or any of the morphemes, there is 40.2% probability that the context is labeled with the proverb '*the more the merrier*'. From this observation, we explore the possibility of adopting tree based classification algorithm into the quote recommendation task.

Among various decision tree algorithms, RF [5] is an ensemble learning method that had notable success in various fields due to its resilience to over-fitting and tendency to exhibit low variance and bias. RF constructs n_{tree} decision trees by training each tree with samples of random subset of features. The method is able to populate each decision tree with the most discriminating quotes at each state and aggregate the results by voting. In the case of our dataset, we view contexts as 'documents' and use bag-of-words TF-IDF as features for each context. Then, we train the Random Forest classifier using the vectors of TF-IDFs and their correct labels i.e. quote. To the best of our knowledge, this is the first time RF classification has been used for quote recommendation.

3.3 Convolutional Neural Network

Word matching-based methods such as context-aware relevance model [1] and citation translation model [3] have difficulty in exploiting n-gram features because of sparsity problem, so they only use unigram-based features. But n-gram features are important because there are many phrases which are meaningful only when the terms in phrase stay together. For example, a phrasal verb *give up* loses the meaning when it is tokenized into *give* and *up*. Unlike matching-based methods, CNN based approach can exploit important n-gram features in the context by learning the parameters of fixed size filters for each n-grams. Generally, CNN is composed of several pairs of convolution layer and max-pooling layer which capture the local patterns from the training example and down-sample extracted features in order to prevent overfitting. When CNN is applied to natural language sentences, it captures the significant local semantics, i.e., n-gram.

We adopted a single-layer CNN, mainly inspired by [4] which reports that simple CNN model shows similar performance to complex one with several convolutions-pooling layers in order to capture distinguished n-gram features in contexts of quotes. Our CNN model takes a context in the form of a list of word embedding vectors of the words in the context. Then the input matrix, a list of context vectors, are fed to the layer which is composed of single convolution layer and max-pooling layer. After that the output vector is fed to fully connected softmax layer in order to compute the probability of candidate quotes and rank the quotes. We use filter size of 3 and 500 hidden nodes in the hidden layer. We also exploit dropout method to prevent overfitting.

3.4 Recurrent Neural Network

We use RNN to tackle our quote recommendation problem in perspective of language modeling, which means that we treat each quote as a special token or word and compute the probability of it given context. While none of above approaches uses order information of words in the context, RNN based approach can model such sequence of words recursively. We use long short-term memory unit (LSTM) [2] which is a recurrent neural network consists of three gates (forget, input, output) those control the networks to learn long-term dependencies without loss of information. The input vector of each time step passes through the three gates and updates latent vectors which LSTM is retaining. In our model we recurrently feed LSTM with a sequence of words in the context in the form of list of word embedding vectors. We use pre-trained word embedding for mapping each word to word vector. The output vector of LSTM layer is passed to fully connected layer and softmax layer in order to compute the probability of target quotes to be recommended. We also use 500 dimension hidden vector in LSTM and also use dropout method.

3.5 Rank Aggregation

We observed that previously proposed algorithms show different recommendation results according to queries (we will

discuss this in the experiment section). This suggests that instead of relying on the single best ranking algorithm, it is better to aggregate rank values of all of the single algorithms to produce accurate and robust ranking, called rank aggregation (RA).

We propose two methods which can aggregate the individual ranking results of previously proposed algorithms. Traditional rank aggregation method Borda [8] assigns a score to candidate quote inversely proportional to its position in a ranked list of individual algorithm, and the scores of each quotes are added up to the final score. We observed that Borda cannot handle the case where one or two inaccurate rank of individual algorithms lowers accuracy of final aggregated rank. In order to cope with this issue, we propose a rank aggregation method called Rank Multiplication (RM) to multiply the ranks of each quotes submitted by individual algorithm. By using this method, we can get the effect that maintaining case that all of the individual ranker rank consistently high, it can give less weight to result of inaccurate ranking algorithm. Thus final score by using RM can be defined as follows:

$$s_{RM}(q,t) = \frac{1}{\prod_i^{|A|} r_i(q,t)}$$

where $r_i(q,t)$ is position in ranked list of ith individual ranking algorithm given query q and candidate quote t. And A is a set of each algorithm. The quotes are ordered by this score in descending order.

We assume that high ranks of individual ranking algorithms are more dependable than lower ranks. From this assumption we propose second rank aggregation method called top-k Rank multiplication (top-k RM) that multiplies only k rank values of a quote from each of k single algorithms for a query. Thus the final score of the top-k RM is defined as follows:

$$s_{TopK\_RM}(q,t) = \frac{1}{\prod_{i \in TopK}^{|A|} r_i(q,t)}$$

where $TopK$ is a set of k algorithms that yield the k highest rank positions given query q and quote t.

4. EXPERIMENTS

4.1 Data Construction

We have collected 439,655 quotes from three sources: *Wikiquote*[1]*, Oxford Concise Dictionary of Proverbs*[2], and *Library of Quotes*[3]. For the context data, we searched blocks of texts that contain these quotes from three different sets of corpus: 2 million tweet threads from Twitter (~2015.11.15), 20GB of electronic book from the *Project Gutenberg Database*[4], and 190GB of ICWSM spinn3r 2009 blog dataset[5]. In the tweet corpus, in order to extract dialogs only, we selected threads where only two users are involved. Next, we chose the top 400 quote set from each corpus according to the number of contexts, in order to reflect the characteristics of the quotes that appeared frequently in different corpus. Finally, we generate three datasets: Twitter dataset, Gutenberg dataset, and Blog dataset.

[1] https://en.wikiquote.org/

[2] Oxford University Press, 1998

[3] http://www.libraryofquotes.com/

[4] http://www.gutenberg.org/

[5] http://icwsm.cs.umbc.edu/data/icwsm2009/

Table 1 number of contexts for each quote in datasets

Datasets	Avg	Std dev	Max	Min
Twitter	556	971	10764	15
Gutenberg	89	122	1366	14
Blog	230	543	5923	24

Table 1 shows the number of context for each quote in each datasets, which describes average, maximum, and minimum number of context for each quote and standard deviation of them. From Table 1, we see that the most frequently appeared quotes from each corpus cover large range of quotes of varying frequencies, helping us deal with the situation recommending quotes by using small number of contexts as well as large number of contexts. We divide dataset to the proportion of 8:1:1, as training set, validation set, and test set. We create test sets by hiding the quotes which the contexts are paired with.

4.2 Evaluation Metric

We consider a plausible application that recommends only a few number of quotes. In such application since the position of correct quote is not important, we use Recall@k as our evaluation metric.

Recall@*k*: Since there is only one correct or hidden quote for each query in the original test set, Recall@k is the number of cases that the gold quote is recommended in the top-k result divided by number of total test cases. We set k as five.

4.3 Baselines and Parameter Settings

We compare our approaches with three state-of-the-art approaches in quote or citation recommendation domain. Learning-to-recommend quote (LRQ) [7] is an algorithm for recommending quote for wring. Context-aware relevance model (CRM) [1], citation translation model (CTM) [3] are algorithms for recommending citation for scientific paper. Also popularity-based method (Popularity), and cosine similarity-based method (Cosine similarity) is adopted as baselines. The methods, Popularity and Cosine similarity methods are used in order to reveal the different levels of difficulties of the datasets. These methods are described in detail below.

LRQ exploits an existing learning-to-rank framework for quote recommendation with quote-based features, quote-query similarity features, and context-query similarity features.

CRM recommends quotes according to average of the squared cosine similarities between contexts of each quote and the query.

CTM recommends quotes according the probability that the query context would be translated into the quote.

Popularity ranks the quotes according to their frequency in contexts of training set.

Cosine similarity ranks the quote by examining individual context of the quote with the given query using bag-of-words representation.

We implement these methods and set the parameters to optimum as specified in the respective papers of the methods. Specifically, we truncate each half-context (pre-context or post-context) of length longer than 150 characters for LRQ, 50 words for CRM and one sentence for CTM respectively as the respective authors suggested in the papers. For our approaches, we set length of half-context to its optimal value which shows best result in

validation dataset: 1) 150 characters of pre-context and post-context with word truncation for context clustering and context lumping, 2) 50 words for RF, and 3) 30 words of pre-context for CNN and RNN. As stated in the introduction, we used pre-context and post-context as query for Gutenberg and Blog dataset and pre-context as query for Twitter dataset. Hyper parameters of single algorithms are set by using validation set. For rank aggregation, we used the proposed five algorithms (context clustering, context lumping, RF, CNN and RNN) and, top-k RM showed best results when k=3.

4.4 Results and Discussions

Results of experiments are listed in Table 2. Recall@5 and the improvement ratio of each algorithm over the best baseline in each dataset are denoted. The individual algorithms (context lumping and CNN), even without rank aggregation, outperform baselines in all of the datasets. Surprisingly, the simple method context lumping is the best performer in Gutenberg and Blog dataset, which beats LRQ up to 35%. Context clustering outperforms CRM and Cosine similarity which does not treat the context of quote collectively. These better results of context lumping and context clustering show the effectiveness of adjusting context matching granularity. One can observe that performance of the baseline Cosine similarity in Twitter dataset is worse than ones in Gutenberg and Blog dataset. This means that sparsity problem is more serious in Twitter where the tweet contains more infrequent words than others. In Twitter dataset, deep learning algorithms (CNN and RNN) outperform CTM by up to 43%. From this result, we can see that deep learning algorithms are able to mitigate such serious sparsity problem because it is not based on word matching. Results of RF show that it is competitive to CTM algorithm. In fact, in our preliminary experiments on top 100 Twitter dataset, RF outperforms CNN. However, in large dataset, generalization of the algorithm is not made as expected; an area for future investigation.

Although some of our single algorithm outperform others in specific datasets, there is no single algorithm that outperforms all the others. Also even in a dataset, there exists a portion of queries where each of single recommendation algorithms is exclusively correct. See Table 3. These justify our motivation of adopting rank aggregation, and as expected, improvement attained through rank aggregations (RM RA and top-k RM RA) are better than the best baseline algorithm on average 44.0% and 46.7% respectively.

Table 2 Results of Recall@5 of different methods.

Context source / Approaches	Twitter	Gutenberg	Blog
Context clustering	0.190 (-30%)	0.299 (- 1 %)	0.494 (0 %)
Context lumping	0.286* (+ 5%)	0.409* (+35%)	0.521* (+ 5 %)
RF	0.244 (- 11%)	0.246 (-19 %)	0.470 (- 5 %)
CNN	0.390* (+43%)	0.326* (+ 8 %)	0.506 (+ 2 %)
RNN	0.389* (+42%)	0.294 (- 3 %)	0.473 (- 4 %)
RM RA	0.424* (+55%)	0.445* (+47%)	0.640* (+30 %)
top-k RM RA (k=3)	**0.436*** (+60%)	**0.451*** (+49%)	**0.648*** (+31 %)
LRQ	0.196	0.302	0.494
CRM	0.119	0.237	0.382
CTM	0.273	0.257	0.441
Popularity	0.156	0.111	0.223
Cosine similarity	0.196	0.248	0.469

(* indicates that each of our algorithms outperform the best baseline algorithm with statistically significant increase at $p < 0.01$ in two-tailed t-tests)

Table 3 number of correct cases of single algorithms in Twitter dataset

Approaches	# correct case (A)	#case exclusively correct (B)	(B) / (A)
Context lumping	6,031(0.286)	1,122	0.186
RF	5,186(0.244)	493	0.095
CNN	8,213(0.390)	935	0.114
RNN	8,191(0.389)	1023	0.125

In conclusion, although some of our single algorithms such as context clustering or RF do not outperform the baselines, there are cases where each single algorithm is able to exclusively answer correctly, which we believe we were able to exploit in our proposed rank aggregation method.

5. CONCLUSIONS

In this paper, we tackled quote recommendation by exploring four single recommendation approaches considering different aspects of the context. And we presented new rank aggregation methods for maximizing performance. Over our datasets, we showed that the proposed algorithm (top-k RM RA) outperforms the best baseline by up to 46.7%. In the future, we plan to extend our research to recommend common phrase which has wider applications in the real world.

6. REFERENCES

[1] He, Q., Pei, J., Kifer, D., Mitra P., and Giles, C. L., 2010. Context-aware citation recommendation. In *Proceedings of the 19th international conference on World wide web* (Raleigh, NC, USA, April 26 - 30). WWW '10. ACM, New York, NY, USA, 421-430. DOI=http://dx.doi.org/10.1145/1772690.1772734

[2] Hochreiter, S. and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation* (Nov. 1997). 1735-1780

[3] Huang, W., Kataria, S., Caragea, C., Mitra, P., Giles, C. L., and Rokach, L. 2012. Recommending citations: translating papers into references. In *Proceedings of the 21st ACM international conference on Information and knowledge management* (Maui, HI, USA, October 29 - November 02, 2012). CIKM '12. ACM, New York, NY, USA, 1910-1914.DOI=http://dx.doi.org/10.1145/2396761.2398542

[4] Kim, Y. 2014, Convolutional Neural Networks for Sentence Classification, In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (Doha, Qatar, October 25-29, 2014). EMNLP '14. ACL

[5] Liaw, A. and Wiener, M. 2002. Classification and regression by randomForest. *R News* (2002). 2(3):18–22

[6] Rangrej, A., Kulkarni, S., and Tendulkar, A. V. 2011. Comparative study of clustering techniques for short text documents. In *Proceedings of the 20th international conference companion on World wide web* (Hyderabad, India, March 28 - April 01, 2011). WWW '11. ACM, New York, NY, USA, 111-112. DOI=http://dx.doi.org/10.1145/1963192.1963249

[7] Tan, J., Wan, X. and Xiao, J. 2015. Learning to recommend quotes for writing. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (Austin Texas, January 25–30, 2015). AAAI'15. AAAI Press, USA, 2453-2459

[8] Young, H. P. 1974. An axiomatization of Borda's rule. *J. Econ. Theory* 9, 1, 43-52

Learning-to-Rank in research paper CBF recommendation: Leveraging irrelevant papers

Anas Alzoghbi Victor A. Arrascue Ayala Peter M. Fischer

Georg Lausen

Department of Computer Science, University of Freiburg, Germany
{alzoghba, arrascue, peter.fischer, lausen}@informatik.uni-freiburg.de

ABSTRACT

Suggesting relevant literature to researchers has become an active area of study, typically relying on content-based filtering (CBF) over the rich textual features available. Given the high dimensionality and the sparsity of the training samples inherent to this domain, the focus has so far been on heuristic-based methods. In this paper, we argue for the model-based approach and propose a learning-to-rank method that leverages publicly available publications' meta-data to produce an effective prediction model. The proposed method is systematically evaluated on a scholarly paper recommendation dataset and compared against state-of-the-art model-based approaches as well as current, domain-specific heuristic methods. The results show that our approach clearly outperforms state-of-the-art research paper recommendations utilizing only publicly available meta-data.

CCS Concepts

•**Information systems** → **Learning to rank; Recommender systems;**

Keywords

Research paper recommendation; Learning-to-Rank; Content-based Recommendation; Model-based user profile

1. INTRODUCTION

Scholars and researchers are confronted with an overwhelming number of newly published research papers in their domain of expertise. Although advantageous in restricting the domain, keyword-based search tools typically available in digital libraries offer a limited help to researchers in locating the relevant content. As a result, researchers need to manually search within unspecific search results to identify paper(s) of interest. This is the situation where recommendaer systems have great potential, and indeed plenty of works adopted different techniques to tackle this problem. A recent extensive survey in this domain [3] identified content-based filtering (CBF) as the predominant approach for research paper recommendation because of the rich textual features available. For learning user profile, almost exclusively the focus was on relevance feedback approaches, building on the assumption that papers appearing in user's preference list have an equal (or a presumed extent) share in the underlying user taste. Thus, user profiles are constructed as aggregation of relevant papers' keywords. Based on the classification suggested by Adomavicius et al. in [1], these approaches are referred to as heuristic-based. In contrast, model-based approaches depend on a learning method to fit the underlying user model (profile). This enables constructing a better modeling of researcher-keywords relation in user profiles. But they require a large body of training data which is not intuitively available in this domain. As a result, little work on applying model-based approaches exists for this problem.

In this paper, we employ pairwise learning-to-rank [4] as a model-based technique for learning user profile. We incorporate both relevant and irrelevant "peer" papers -papers published in relevant papers' conferences- to formulate pairwise preferences and enrich the training set. Our main contributions include:

- We investigate and customize learning-to-rank for CBF research paper recommendation.
- We incorporate only a small set of data, restricted to publicly available metadata of papers. This makes our approach suitable for a much larger domain than previous approaches which require papers' full-text.
- We perform an initial, yet systematic study on a real-world datatset in which we show that our approach clearly outperforms existing heuristic- and model-based algorithms.

The rest of this paper is organized as following: the second section provides an overview of existing related work. In section 3 we present our approach and in section 4 we demonstrate experimental setup and results. Finally, we conclude in section 5 by summarizing our findings and situate this work within our future plan.

2. RELATED WORK

A rich amount of related work tackled the problem of research paper recommendation. collaborative filtering (CF) approaches [8, 13, 14] showed a successful application of

CBRecSys 2016, September 16, 2016, Boston, MA, USA.

model-based methods incorporating knowledge from other "similar" users. However, we restrict our search to content-based scenarios considering only information from the active user. In this domain, the main focus in learning user profile has been on heuristic-based approaches with a wide adoption of relevance feedback and cosine similarity [3]. Papers are recommended which are most similar to one or more of previously published or liked papers. In [10], De Nart et al. used extracted terms (keyphrases) from user's liked papers in constructing user profile. The profile has a graph representation, and the focus here was on the keyphrases extraction method and the graph structure. The approach of Lee et al. [6] proposed a memory based CBF, where users' papers are clustered based on their similarity, and candidate papers are ranked based on the distance from user's clusters. Sugiyama et al. in [11, 12] applied a relevance feedback approach utilizing all terms from the fulltext of the researcher's publications in addition to terms from the citing and the referenced papers in order to build profiles. All of these works are heuristic-based, where weights in user profile are set by aggregating individual keywords' scores of relevant papers. On the contrary, model-based approaches depend on machine learning techniques to learn user affinity towards keywords, promising a more representative user profile. In a previous work [2], we showed the superiority of a model-based method over relevance feedback methods for CBF research paper recommendations. We applied multivariate linear regression to learn researchers' profiles from their previous publications. Yet, the work was tailored to researchers with previous publications and didn't consider irrelevant papers. In [9], Minkov et al. presented a collaborative ranking approach for events recommendation. They compared it with a content-based baseline that applies pairwise learning-to-rank on pairs of relevant and irrelevant events. In our work, we follow similar approach in applying learning-to-rank on pairs of relevant an irrelevant papers. However, we push it further and investigate the quality of these pairs and their effect on the model performance.

3. PROPOSED APPROACH

This work targets users who have previously interacted with scientific papers and identified some as papers of interest (relevant papers). Having a set of relevant papers for a user, the recommendation process can start and a machine learning method is applied to fit a user profile (model). The learned model is used to rank a set of candidate papers and recommend the top ranked papers to the user. Our approach is to employ the pairwise learning-to-rank technique in building the user profile. We chose this method because of its desirable properties: It was proven to be successful in solving ranking tasks in similar problem domains like online advertising [7]. It also shows a good performance on problems with sparse data. The main idea of pairwise learning-to-rank is to build pairs of preferences out of the training set. Each pair consists of a positive and a negative instance. Afterwards, the pairs are fed as training instances to a learning algorithm, which in turn learns the desirable model. In the underlying problem, papers marked as interesting by users are the positive instances. However, the negative instances or the irrelevant papers are usually not explicitly provided by the users. This makes pairwise learning-to-rank not directly applicable on this setup. In our contribution, we seek implicit information about the irrelevant papers. For this, we start from the following hypothesis: when users identify relevant papers, they, to some extent, implicitly rate other papers published at the same conference (we call them *peer papers*) as irrelevant[1]. Based on this hypothesis, we utilize peer papers as irrelevant papers as follows: for each user, we build pairs of preferences out of relevant and peer papers. Such pairs are called pairwise preferences or for simplicity pairs, we will use these terms interchangeably along the paper. Afterward, we feed these pairs as training examples to a learning algorithm in order to fit the user's model. This model is used later to rank candidate papers and recommend top ranked ones to the user. Before delving deeper in the method details, we first introduce some notation. The function $peer(.)$ is defined over the interest set P^r_{int} of a user r. It delivers for a paper $p \in P^r_{int}$ the set of p's peer papers. In practice, this can be retrieved via digital libraries like DBLP registry[2]. For the paper modeling, we adopt a vector space model representation. Having the domain related keywords extracted from paper's title, abstract and keyword list as features, each paper p is a vector: $p = \langle s_{p,v_1}, ..., s_{p,v_{|V|}} \rangle$, with $v_i \in V$ is a domain-related vocabulary and s_{p,v_i} is a score reflecting the importance of v_i in p. We adopt the TF-IDF score as the weighting scheme. Based on this representation, the similarity between two papers is calculated by the cosine similarity between the papers' vectors.

3.1 Method Steps

An overview of the proposed approach is depicted in Figure 1. For the experimental setup only, we split user's r interest set P^r_{int} into training and test sets P^r_{train}, P^r_{test} respectively. However, this step is dropped out in the non-experimental recommendation scenario and the first step receives, in this case, the complete interest set P^r_{int}.

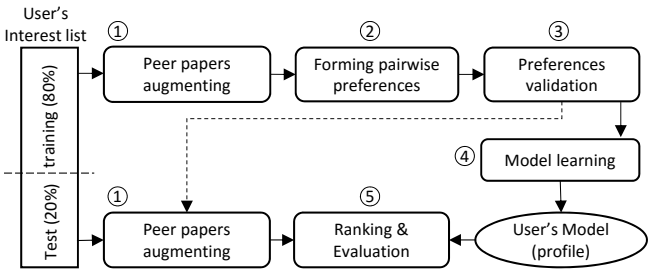


Figure 1: Overview the proposed approach steps

1. Peer papers augmenting: in this step, the peer papers are retrieved for all relevant papers. Retrieved peer papers serve as potential negative classes and are important for empowering the learning algorithm to construct a better understanding of user's taste.
2. Forming pairwise preferences: here we apply the concept of pairwise learning from learning-to-rank. The training set in this step is reformulated as a set of pairs $\mathcal{P}$, where each pair consists of two components: a relevant paper and an irrelevant paper. That is, each relevant paper $p \in P^r_{train}$ is paired with all papers from $peer(p)$:

$$\mathcal{P} = \{(p, p') | \forall p \in P^r_{train} \wedge \forall p' \in peer(p)\}$$

[1]Later, we introduce a validation process that checks the correctness of this hypothesis for each pair.
[2]http://dblp.uni-trier.de

A pair $(p, p') \in \mathcal{P}$ depicts a preference in user's taste and implies that p has a higher relevance to user r than p'.

3. Preferences validation: In the first step, we introduced the peer papers as negative classes based on the hypothesis mentioned earlier in this section. Yet, this can't be adopted as a ground truth due to: (a) it is not explicitly affirmed by users that they are not interested in peer papers; and (b) some peer papers might be of interest to the user but might have been overlooked. Having this in mind, not all pairwise preferences formulated in the previous step have the same level of correctness. Therefore, this step examines pairwise preferences and makes sure to pass valid ones to model learning. We propose two different mechanisms to accomplish this validation: pruning based validation and weighting based validation. We explain these techniques in the next section.
4. Model learning: In this step, we apply a pairwise learning-to-rank method (Ranking SVM [5]) to train a user model $\hat{w}_r$. Using validated pairwise preference from the previous step, we seek $\hat{w}_r$ that minimizes the objective function:

$$\hat{w}_r = \arg\max_{w_r} \frac{1}{2}||w_r||^2 + C.\mathcal{L}(w_r)$$

With $C \geq 0$ is a penalty parameter and $\mathcal{L}(w_r)$ is the pairwise hinge loss function:

$$\mathcal{L}(w_r) = \sum_{(p,p') \in \mathcal{P}} max(0, 1 - w_r^T(p - p'))^2 \qquad (*)$$

5. Ranking & Evaluation: Given the user's model as a result of the previous step, here we apply the prediction on candidate papers. For the experimental setup, this is the the test set, which is constructed out of relevant papers P^r_{test} (the positive instances), in addition to their peer papers as irrelevant papers (the negative instances).

3.2 Preferences Validation Methods

As pairwise learning-to-rank expects pairs that show contrast between negative and positive classes, pairs with "wrongly assigned" peers pose a potential noise to the learning process. After all, the validity of a pairwise preference (p, p') depends on the correctness of considering its peer paper p' irrelevant. The pair's relevant paper p forms the ground truth and hence, it can be considered as the reference point to decide whether p' is irrelevant or not. For each pair $(p, p') \in \mathcal{P}$ we measure the similarity between p and p', and adopt two methods to validate the pair based on this similarity:

Weighting Based Validation (WBV). This strategy is based on giving pairwise preferences different weights based on the dissimilarity between the pairs components. This boosts the importance for pairs with dissimilar components and assures that the more similar the pair's components are, the less important the pair for model learning is. Therefore, we weight the importance of each pair according to the distance (1-similarity) between the relevant paper and the peer paper. Then, we redefine the loss function from (*) to consider pairs' weights as following:

$$\mathcal{L}(w_r) = \sum_{(p,p') \in \mathcal{P}} max(0, 1 - w_r^T(1 - similarity(p, p'))(p - p'))^2$$

Pruning Based Validation (PBV). Here we filter out invalid pairwise preferences. Validity is judged based on the dissimilarity between the pair's components. If they prove to be similar, then we don't consider p' as an irrelevant paper and consequently, the pair (p, p') is not eligible for model learning. A similarity threshold τ is applied and a pair (p, p') is pruned if $similarity(p, p') > \tau$. In our experiments, we empirically test a range of values for τ and discuss the corresponding effect on the model.

4. EXPERIMENTS

4.1 Dataset & Setup

We evaluated the proposed approach on the Scholarly publication recommendation dataset from [12], including the extensions applied in our previous work [2]: Papers are identified and enriched with meta-data from the DBLP register, namely titles, abstracts, keywords and the publishing conference. The dataset contains 69,762 candidate papers, as well as the lists of relevant papers for 48 researchers. The number of relevant papers ranges from 8 to 208 with an average of 71 papers. After augmenting peer papers, we got a skewed distribution as the ratio of relevant papers to peer paper ranges from 0.45% to 3% with an average of 1.2%. We performed offline experiments with 5-folds cross validation following the steps outlined in Figure 1. For each researcher we randomly split the interest list into training and test sets; then, we learn researchers' models as described in section 3; finally, we evaluate the learned models on the test set. The test set consists of: (a) positive instances, the test relevant papers (20% of the researchers interest list) and (b) negative instances, the peer papers of the positive instances. This applies for all of our experiments, except for experiments on the pruning based validation method (PBV). In PBV, we filter out those pairs which components have a similarity higher than τ from the training set. Therefore, we apply the same rule on the test set and we filter out peer papers based on their similarity to the corresponding relevant paper. For example, given a similarity threshold τ and a relevant paper p from the test set, a peer paper $p' \in peer(p)$ is added as an irrelevant paper to the test set if and only if $similarity(p, p') \leq \tau$.

4.2 Metrics

We measured the following metrics to determine the performance for top k ranking and also overall classification. We show the averages over all researchers for each metric:

Mean Reciprocal Rank (MRR): evaluates the position of the first relevant paper in the ranked result.

Normalized Discounted Cumulative Gain (nDCG): nDCG@k indicates how good the top k results of the ranked list are. We look at nDCG for $k \in \{5, 10\}$

AUC and Recall: used to study the behavior of validation strategies PBV, WBV and the baseline algorithms: Logistic Regression and SVM.

4.3 Results & Discussion

In total, we performed three different experiments. The first experiment (with the results shown in Table 1) shows a superior performance for our weighting based validation method (WBV) over the state-of-the-art heuristic-based work (Sugiyama [12]) and model-based (PubRec [2]) approach.

The experiments were performed using the same features and datasets present in these works and show a clear lead over all metrics.

	MRR	nDCG@5	nDCG@10
WBV	0.728	0.471	0.391
PubRec	0.717	0.445	0.382
Sugiyama[12] via [2]	0.577	0.345	0.285

Table 1: WBV compared to state-of-the-art model-based and heuristic-based approaches

The second experiment compares the performance of our approach over other, baseline classification algorithms like SVM and logistic regression to provide a more general understanding of its capabilities. As shown in Figure 2, logistic regression showed a weak performance on all metrics, particularly on Recall. It didn't succeed in identifying relevant papers even when it is fed with a balanced training set. However, SVM showed a better ability to recognize the relevant papers with a better recall value, but produced a lot of false positives and this is clear from its lower MRR and nDCG values. In contrast, all variants of our method showed a superior performance in all metrics. Finally, we compare between the suggested pair validation techniques WBV and PBV, including tuning the latter by varying the similarity threshold τ from 1 (where no pairs are filtered, this case represents the CBF approach of [9]), down to $4 * 10^{-4}$ (where a lot of "noisy" pairs are pruned from the training set). WBV showed in general a very good performance, beating PBV for higher values of τ on all metrics except recall. There, PBV gives a slightly better recall even without filtering any pairs (when $\tau = 1$). This refers to the fact that weighting the pairs in WBV causes the model to miss some relevant papers, while PBV made models more capable of recognizing the relevant papers by eliminating the noisy pairs from the training set. When decreasing τ, PBV shows very good scores, but these results need additional investigation before leading to a clear conclusion. As mentioned earlier in this section, reducing τ also leads to a smaller number of irrelevant papers in the test set. This reduces the underlying bias in the test set which has an (additional) positive impact on the metrics, even though there is still a clear bias (the relevant/peer ratio is on average 11.2%) present at the lowest τ values.

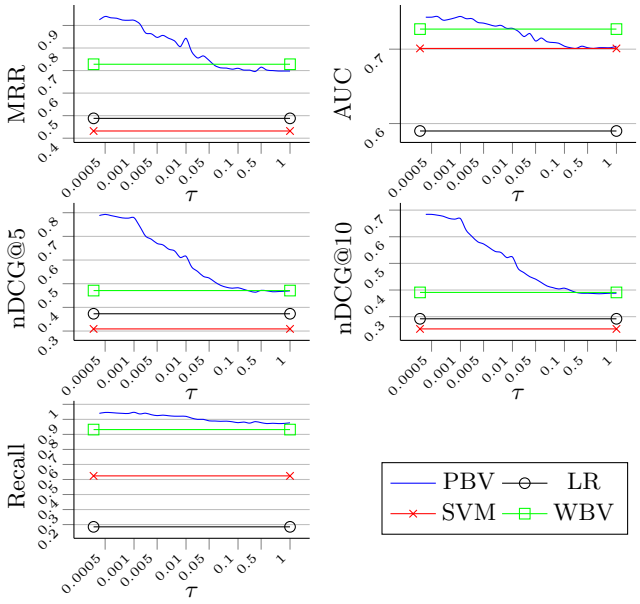


Figure 2: WBV and PBV compared with Logistic regression and Support Vector Machine

5. CONCLUSION

In this paper, we investigated the application of learning-to-rank in research paper recommendation. We proposed a novel approach that leverages irrelevant papers to produce more accurate user models. Offline experiments showed that our method outperforms state-of-the-art CBF research paper recommendations utilizing only publicly available metadata. Our future steps will focus on further understanding the effect of the similarity threshold in pruning based validation (PBV) on the model quality and study the suitability of pairwise learning-to-rank algorithms other than Ranking SVM for this problem.

6. REFERENCES

[1] G. Adomavicius, Z. Huang, and A. Tuzhilin. *Personalization and Recommender Systems*. 2014.

[2] A. Alzoghbi, V. A. A. Ayala, P. M. Fischer, and G. Lausen. Pubrec: Recommending publications based on publicly available meta-data. In *LWA*, 2015.

[3] J. Beel, B. Gipp, S. Langer, and C. Breitinger. Research-paper recommender systems: a literature survey. *IJDL*, 2015.

[4] L. Hang. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 2011.

[5] R. Herbrich, T. Graepel, and K. Obermayer. *Large Margin Rank Boundaries for Ordinal Regression*. 2000.

[6] J. Lee, K. Lee, J. G. Kim, and S. Kim. Personalized academic paper recommendation system. In *SRS*, 2015.

[7] C. Li, Y. Lu, Q. Mei, D. Wang, and S. Pandey. Click-through prediction for advertising in twitter timeline. In *KDD*, 2015.

[8] S. M. McNee and et al. On the recommending of citations for research papers. In *CSCW*, 2002.

[9] E. Minkov, B. Charrow, J. Ledlie, S. Teller, and T. Jaakkola. Collaborative future event recommendation. CIKM, 2010.

[10] D. D. Nart and C. Tasso. A personalized concept-driven recommender system for scientific libraries. *Procedia Computer Science*, 2014.

[11] K. Sugiyama and M.-Y. Kan. Scholarly paper recommendation via user's recent research interests. In *JCDL*, 2010.

[12] K. Sugiyama and M.-Y. Kan. Exploiting potential citation papers in scholarly paper recommendation. In *JCDL*, 2013.

[13] A. Vellino. A comparison between usage-based and citation-based methods for recommending scholarly research articles. In *ASIS&T*, 2010.

[14] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *KDD*, 2011.

Recurrent Neural Networks for Customer Purchase Prediction on Twitter

Mandy Korpusik
Computer Science & Artificial Intelligence Laboratory, MIT
Cambridge, Massachusetts
korpusik@mit.edu

Shigeyuki Sakaki
Fuji Xerox Co., Ltd., Japan
sakaki.shigeyuki@fujixerox.co.jp

Francine Chen Yan-Ying Chen
FX Palo Alto Laboratory, Inc.
Palo Alto, California
{chen, yanying}@fxpal.com

ABSTRACT

The abundance of data posted to Twitter enables companies to extract useful information, such as Twitter users who are dissatisfied with a product. We endeavor to determine which Twitter users are potential customers for companies and would be receptive to product recommendations through the language they use in tweets after mentioning a product of interest. With Twitter's API, we collected tweets from users who tweeted about mobile devices or cameras. An expert annotator determined whether each tweet was relevant to customer purchase behavior and whether a user, based on their tweets, eventually bought the product. For the relevance task, among four models, a feed-forward neural network yielded the best cross-validation accuracy of over 80% per product. For customer purchase prediction of a product, we observed improved performance with the use of sequential input of tweets to recurrent models, with an LSTM model being best; we also observed the use of relevance predictions in our model to be more effective with less powerful RNNs and on more difficult tasks.

CCS Concepts

•**Information systems** → **Social recommendation; Personalization;** Social networks;

Keywords

Deep learning, Recommender systems, Microblogs

1. INTRODUCTION

In social media, popular aspects of customer relationship management (CRM) include interacting with and responding to individual customers and analyzing the data for trends and business intelligence. Another mostly untapped aspect is to predict which users will purchase a product, which is useful for recommender systems when determining which users will be receptive to specific product recommendations. Many people ask for input from their friends before buying a higher-priced product, and users will often turn to social media for that input [13]. Many users also post announcements about significant purchases. Thus, social media posts may contain cues that can be used to identify users who are likely to purchase a product of interest as well as later post(s) indicating that a user made a purchase.

Looking to buy a camera for X-mas, can I get some help?
Mann wish I had an iphone,I wanna be cooolll too! #sadtweet
Now thinking of getting a new phone
My baby arrived #panasonic #lumix - its waterproof but I daren't try it hah :)
Got a #Xoom tablet today. Already rooted and master of my domain. Thanks @koush

Table 1: Sample tweets indicating a user wants to buy (top three) and bought (bottom two) a product.

In this paper, we present our investigations of deep learning methods for predicting likely buyers from microblogs, specifically Twitter tweets. While many social media posts are private, semi-private or transient, and thus hard to obtain, microblogs such as tweets are generally public, simplifying their collection. With the ability to identify likely buyers, as opposed to targeting anyone who mentions a product, advertisements and products can be presented to a more receptive set of users while annoying fewer users with spam.

Microblogs cover a variety of genres, including informative, topical, emotional, or "chatter." Many of a user's tweets are not relevant, that is, indicative of whether a user is likely to purchase a product. Thus, we hypothesize that identifying tweets that are relevant to purchasing a product is useful when predicting whether a user will purchase that product. We also hypothesize that a sequence of tweets contains embedded information that can lead to more robust prediction than classification of individual tweets. For example, with the given order of the second and third tweets in Table 1, the user seems interested in buying a new phone, but if the order is reversed, the inference is that the user was thinking of buying a phone, but did not buy one. In this work, we

- investigate the use of recurrent neural networks to model sequential information in a user's tweets for purchase behavior prediction. Our use of recurrent models enables previous tweets to serve as context.
- introduce relevance prediction into the model for reducing the influence from noisy tweets.

This paper is organized as follows. In the next section, we describe related work in deep learning and social media product interest prediction. Afterward, we detail our data

CBRecSys 2016, September 16, 2016, Boston, MA, USA.

Tweet Type	Regular Expression
bought	"my new .* X", "gotta new .* X", "bought a .* X", "splurged on a .* X"
want	"should I buy .* X", "wanna .* X", "should I go for .* X", "need a new .* X"

Table 2: Sample of expressions used to identify candidate bought/want users, where X is a product.

Final User Label	Candidate Want User	Candidate Buy User
Buy User	64	2491
Not-Buy User	1226	315

Table 3: Corpus statistics of annotated candidate users collected via tweets containing want/buy expressions that were then labeled buy/not-buy.

collection and annotation process. Then, we explain the deep learning methods, discuss the experiments, and analyze results. Finally, we conclude and propose future work.

2. RELATED WORK

There are several works related to identifying customers with interest in a product. A system developed by [14] for predicting user interest domains was based on features derived from Facebook profiles to predict from which category of products an eBay user is likely to make a purchase. [9] used a rule-based approach for identifying sentences indicating "buy wishes" from forums with buy-sell sections. [3] presented a method for predicting whether a single post on Quora or Yahoo! Answers contained an expression of "purchase intent." However, some of their Purchase Action words, including "want" and "wish" only indicate interest in a product. Although users may say they want a product, many of these Twitter users will not buy the product in the near future (see Figure 2 and Table 3). For our task, we predict whether a user will actually make a purchase; this is different from the task of predicting user interest in a product. Often users with interest in a product may not have the means to purchase soon or may say they want or need something as an indication that they like something.

Our approach also differs from these works in that access to Facebook profiles is not needed, and features are automatically learned using neural networks. In addition, most of the earlier works classify a single sentence or posting. In contrast, we predict user behavior based on past postings, that is, from a *sequence of tweets*, which enables preceding tweets in a sequence to provide context for the current tweet.

Our approach draws on the deep learning Recurrent Neural Network (RNN) models which have been successfully applied to sequential data such as sentences or speech. For example, [5] used a combination of a convolutional neural network to represent sentences and an RNN to model sentences in a discourse. [2] observed that long short-term memory (LSTM) models, a type of RNN with longer memory, performed better than an RNN on the TIMIT phoneme recognition task. In our work, we compare RNN and LSTM models for the task of predicting whether a user will buy a product of the type that they have mentioned.

3. DATA COLLECTION AND LABELING

For the buy prediction task, we focused on two product categories: (1) cameras and (2) mobile devices, i.e., mobile phones, tablets, and smart watches. These are generally higher-priced products which users do not purchase frequently and therefore are more likely to tweet about.

We created a separate corpus for each category composed of tweets by users who either: (1) bought a target product or (2) wanted, but did not buy, a target product [10]. To collect tweets by each user, we first identified from eBay listings a set of model names for each product category. Similar model names were merged, e.g., "iPhone4" and "iPhone5" into "iPhone," resulting in 146 camera names and 80 mobile device names. We also created a set of regular expressions that may indicate a user bought or wanted one of the products (see sample in Table 2). Tweets containing a bought or want expression for one of the product names were then collected using the Twitter search API, and the user of each tweet was identified from the tweet meta-data. The tweets of the identified users were collected using the Twitter search and timeline APIs. We called users found with "bought" regular expressions *candidate buy users*, and users identified from "want" regular expressions *candidate want users*.

Due to poor labeling performance by Mechanical Turkers, who often were not familiar with many of the lesser-known mobile devices and cameras, we used an "expert" annotator to whom we gave many examples of labeled bought and want tweets, including trickier cases. For example, a user did not buy a camera if they were given one or if they retweeted (RT) a user who bought a camera. A sample of the labels was checked for accuracy by one of the authors. The annotator determined whether each *candidate want user* tweeted that they bought the product type of interest; if so, the *candidate want user* was labeled a *buy user* (Table 3). Similarly, the tweets of each *candidate buy user* were examined for at least one tweet indicating that the user really bought the target product type. In total, we annotated tweets from 2,403 mobile device users and 1,252 camera users. The annotator also labeled a separate random sample of tweets as relevant/not to predicting whether a target product was bought.

4. DEEP LEARNING METHODS

In this section, we describe the neural network (NN) models we implemented in Python's Theano toolkit [1] for classifying tweets as relevant/not and predicting whether a Twitter user bought a product 60 days after tweeting about it.

The logistic regression (**LR**) model combines the input with a weight matrix and bias vector, feeding it through a softmax classification layer that yields probabilities for each class i. The class i with the highest probability is the output.

A feed-forward (**FF**) network enables more complex functions to be computed through the addition of a sigmoid hidden layer below the softmax. A natural extension of the **FF** network for sequences is a recurrent neural network (**RNN**), in which the hidden layer from the previous timestep is fed into the current timestep's hidden layer:

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b) \quad (1)$$

where h_t is the hidden state, x_t is the input vector, W is a learned weight matrix, and b is a learned bias vector.

Thus, information from early words/tweets is preserved across time and is still accessible upon reaching the final word/tweet for making a prediction. However, typically the error gradient vanishes as the sequence becomes increasingly long, which in practice causes information loss over long time

spans. To compensate for this, long short-term memory [4] uses input, output, and forget gates to control what information is stored or forgotten within a memory cell over longer periods of time than a standard RNN. In the **LSTM**, the input gate i_t, forget gate f_t, and candidate memory cell $\widetilde{C_t}$ are computed using input x_t and previous hidden layer h_{t-1}, weight matrices (i.e., W_i and U_i for the input gate, W_f and U_f for the forget gate, and W_c and U_c for candidate $\widetilde{C_t}$), and forget gate bias term b_f as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1}) \quad (2)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (3)$$

$$\widetilde{C_t} = \tanh(W_c x_t + U_c h_{t-1}) \quad (4)$$

The new memory cell C_t is generated from the combination of the candidate memory cell $\widetilde{C_t}$, controlled by the input gate i_t through element-wise multiplication, and the previous memory cell C_{t-1}, modulated by the forget gate f_t:

$$C_t = i_t * \widetilde{C_t} + f_t * C_{t-1} \quad (5)$$

The output gate and new hidden layer are computed by:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t) \quad (6)$$

$$h_t = o_t * \tanh C_t \quad (7)$$

To preprocess the data, each tweet was first tokenized by the TweeboParser [6], a dependency parser trained on tweets. Then each token was converted into a vector representation using word2vec [7], which learns an embedded vector representation from the weights of a neural network trained on Google News. We also experimented with the GloVe word embedding [8] and with learning an embedding from random initialization. In preliminary experiments on predicting tweet relevance, word2vec consistently performed best and so was used for all reported results.

4.1 Models for Predicting Tweet Relevance

To predict tweet relevance, we compared the LR, FF, RNN, and LSTM models. Since the RNN and LSTM are sequential, the input was an array of token embedded vectors; for the LR model and FF networks, we summed the token embedded vectors as input. For regularization of the LR and FF models, we employed an early stopping technique, and for the RNN and LSTM networks, we incorporated dropout.

4.2 Models for Predicting Purchase Behavior

To predict whether a user will buy a product based on their tweets, we propose a configuration of neural networks that uses predicted tweet relevance in purchase prediction. The input for each user is a sequence of *tweets* (instead of words, as is more commonly used) enabling the preceding tweets to provide context for the current tweet. To model the information in a tweet sequence, a recurrent network (e.g., RNN or LSTM) is intuitively a good choice.

In our proposed joint model (Figure 1), tweets from a user are input as a sequence where each tweet is represented as the sum of the embedded vectors representing its words. The (optional) lower sub-network predicts the relevance of each tweet; we use the best of the four types of relevance classification models, the feed-forward neural net.

The buy sub-network at each time step (i.e., tweet) is composed of either an RNN or an LSTM memory cell, which is fed a tweet vector along with its predicted relevance. The maximum across each dimension of all the tweets' hidden layer outputs are fed into the softmax classifier for the final buy/not buy prediction. The softmax will generalize well to future work on predicting other labels besides buy/not-buy.

For all experiments, each data set was split into 10 partitions for 10-fold cross-validation. Within each fold, 10% was for validation, 10% for testing, and the rest for training. In order to incorporate the FF classifier for predicting tweet relevance as a sub-network in a joint network, we trained a separate classifier for each of the 10 cross-validation partitions. We selected all the users in the training and validation sets for that partition, and trained the relevance classifier on all those users' tweets for which we had a relevance label. The predicted relevance for each tweet was used as an additional feature to predict the user's purchase behavior.

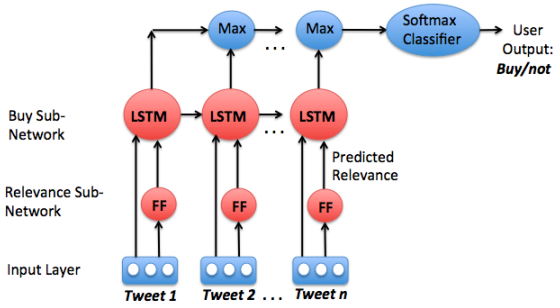


Figure 1: Purchase behavior prediction network.

5. EXPERIMENTS AND RESULTS

We conducted two sets of experiments: first, predicting whether a tweet is relevant to a user's purchase behavior; second, predicting whether a Twitter user will eventually purchase a product within 60 days of tweeting about it.

For all the networks, we set the hidden layer size to 50. The weight matrices were initialized randomly, and bias vectors were initialized to zero. We used RMSprop [12] and negative log-likelihood for training, sigmoid nonlinear activations, and batches of size 10 for up to 100 epochs.

5.1 Tweet Relevance

We observe from the results shown in Table 4 that the FF model performed best. The task is harder for combined mobile device and camera data than for the two individual products, likely due to differences between domain-specific relevance indicators for cameras versus mobile devices.

Model	Mobile	Camera	Both
Logistic	79.7	78.8	74.7
FF	**81.2**	**80.4**	**78.0**
RNN (25%)	80.1	79.2	77.7
LSTM (50%)	80.2	77.0	77.0

Table 4: Accuracy of learning models for tweet relevance. The best dropout rates are 25% and 50% [11].

5.2 User Purchase Behavior

We explored RNNs and LSTMs for predicting whether or not a Twitter user would buy a product, since these models sequentially scan through a user's tweets. We incorporated the FF tweet relevance prediction model as a sub-network in

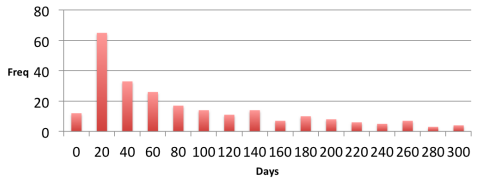


Figure 2: Histogram of the number of days between the first "want" and "bought" tweets by "buy" users.

a deep network; this model predicts a feature indicating each tweet's relevance, which is appended to the input tweets.

For each Twitter user, we used all tweets containing a product mention within a 60-day span, limited to users who wrote between five and 100 product-related tweets; the upper limit was used to filter out advertisers. The 60 days are motivated by the assumption that companies are not interested in promoting products for longer periods, and by Figure 2, which shows that about half the users purchase what they want within 60 days.

We evaluated our models on mobile devices, cameras, and the two combined. Negative examples included Twitter users who wanted a product but did not mention buying it. We trained on their tweets from within the 60-day window before their most recent tweet that mentions wanting a product. Positive examples included users who eventually tweeted about buying a product (the *buy users*), but did not include the "bought" tweet or any tweets written afterward.

The 10-fold cross-validation results averaged over several runs (due to random initialization) on the expert-labeled training data are shown in Table 5. As the baseline, we trained a FF model with sums of all tokens across all tweets for each user (*-sum) as the input. We observe better performance when tweet information is input sequentially to a model with memory (*-seq) than when the sequence information is lost by summing (*-sum) tweets. That is, it is important to capture information embedded in the sequence of tweets. As expected, the LSTM consistently outperformed the RNN because it has the ability to retain information over longer time spans. We also observe that the addition of predicted relevance probabilities to the RNN model (*+Rel) improved performance over the simple RNN; however, adding tweets' relevance only improved the LSTM's performance for the harder combined product task, which may indicate that the vanilla LSTM is powerful enough to learn which tweets are relevant or not when trained on a single product.

Model	Mobile	Camera	Both
FF-sum	73.6	66.3	73.4
RNN-seq	80.8	78.0	79.3
RNN-seq+Rel	81.3	80.5	80.1
LSTM-seq	**83.9**	**81.4**	81.5
LSTM-seq+Rel	83.8	80.9	**81.7**

Table 5: Purchase prediction cross-validation.

6. CONCLUSION

In this work, we investigated deep learning techniques for predicting customer purchase behavior from Twitter data that recommender systems could leverage. We collected a labeled corpus of buy/not buy users and their tweets.

A FF neural network performed best at predicting whether a tweet is relevant to purchase behavior, with an accuracy of 81.2% on mobiles devices and 80.4% on cameras. We found that the use of a deep learning model that incorporates sequential information performed better than ignoring sequential information for the purchase prediction task.

Our initial work in this area has many possible extensions. While we used a 60-day window, it would be interesting to observe user purchase probability changes over time. We will also predict related behaviors, e.g., product comparison.

7. REFERENCES

[1] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: New features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[2] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Proc. of ICASSP*, pages 6645–6649. IEEE, 2013.

[3] V. Gupta, D. Varshney, H. Jhamtani, D. Kedia, and S. Karwa. Identifying purchase intent from social posts. In *Proc. of ICWSM*. AAAI, 2015.

[4] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[5] N. Kalchbrenner and P. Blunsom. Recurrent convolutional neural networks for discourse compositionality. In *Proc. of the 2013 Workshop on Continuous Vector Space Models and their Compositionality*. ACL, 2013.

[6] L. Kong, N. Schneider, S. Swayamdipta, A. Bhatia, C. Dyer, and N. A. Smith. A dependency parser for tweets. In *Proc. of EMNLP*. ACL, 2014.

[7] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*, pages 3111–3119, 2013.

[8] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *Proc. of EMNLP*. ACL, 2014.

[9] J. Ramanand, K. Bhavsar, and N. Pedanekar. Wishful thinking finding suggestions and 'buy' wishes from product reviews. In *Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, page 54, 2010.

[10] S. Sakaki, F. Chen, M. Korpusik, and Y. Chen. Corpus for customer purchase behavior prediction in social media. *LREC*, 2016.

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[12] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 2012.

[13] X. Wang, C. Yu, and Y. Wei. Social media peer communication and impacts on purchase intentions: A consumer socialization framework. *Journal of Interactive Marketing*, 26(4):198–208, 2012.

[14] Y. Zhang and M. Pennacchiotti. Predicting purchase behaviors from social media. In *Proc. of WWW*, pages 1521–1532. ACM, 2013.